Maestría en Ingeniería del Software - Arquitectura y Diseño de Software

Proyecto de Investigación Aplicada

Elección de un lenguaje visual de modelado para el diseño de una arquitectura de software sobre una experiencia interactiva de GodiTours Costa Rica

Juan Bautista González Núñez

Febrero, 2024

Declaratoria de Derechos de Autor

Yo Juan Bautista González Núñez, estudiante de la Universidad Cenfotec de Costa Rica, declaro bajo la fe de juramento y consciente de las responsabilidades penales de este acto, que soy autor intelectual de la tesis o proyecto titulado “Elección de un lenguaje visual de modelado para el diseño de una arquitectura de software sobre una experiencia interactiva de GodiTours Costa Rica”.

Por lo que libero a la Universidad Cenfotec de Costa Rica, de cualquier responsabilidad en caso de que mi declaración sea falsa.

Juan Bautista González Núñez
Cédula 1-1463-0467

Agradecimientos

Quisiera que estas líneas sirvan para expresar el agradecimiento a cada una de las personas que me apoyaron y colaboraron en la realización de este proyecto. Especialmente a mi tutor Msc. Ignacio Rejos Zelaya, por su dedicación y orientación a lo largo de la realización de este proyecto, a GodiTours que además de ser amigos en las aventuras extremas que hemos compartido ha sido la empresa que me abrió las puertas para poder implementar este trabajo sin ninguna objeción. A las lectoras Msc. Jirley Ramírez Cordero y Msc. Ericka Solano Fernández por tomarse el tiempo para poder concluir este proceso.

Además, a las personas que fueron participes activas dentro de las diferentes etapas de este trabajo, ya que sin dudarlo dedicaron su tiempo para ayudarme a obtener información necesaria para realizar mis diferentes análisis.

Por último, pero no menos importante, quiero agradecer a mi madre, mi novia y a mi perrita Mabel, que son parte del motor que haya podido terminar este trabajo.

A todos ellos, muchas gracias.

Aprobación del Proyecto

Universidad Cenfotec
Carrera de Postgrado
Maestría Profesional en Ingeniería del Software

TRIBUNAL EXAMINADOR

Este proyecto fue aprobado por el Tribunal Examinador de la carrera: **Maestría Profesional en Ingeniería del Software con énfasis en Arquitectura y Diseño de Software,** requisito para optar por el título de grado de **Maestría,** para el estudiante: **González Núñez Juan Bautista.**

IGNACIO TREJOS ZELAYA (FIRMA)
Firmado digitalmente por IGNACIO TREJOS ZELAYA (FIRMA)
Fecha: 2024.03.18 22:30:03 -06'00'

M.Sc. Ignacio Trejos Zelaya
Tutor

JIRLEY RAMIREZ CORDERO (FIRMA)
Digitally signed by JIRLEY RAMIREZ CORDERO (FIRMA)
Date: 2024.03.21 17:59:41 -06'00'

M.Sc. Jirley Ramírez Cordero
Lector 1

ERIKA PATRICIA SOLANO FERNANDEZ (FIRMA)
Firmado digitalmente por ERIKA PATRICIA SOLANO FERNANDEZ (FIRMA)
Fecha: 2024.03.27 15:18:18 -06'00'

M.Sc. Ericka Solano Fernández
Lector 2

San José, Costa Rica, 6 de marzo de 2024

Firmada digitalmente, de conformidad con la Ley de Certificados, Firmas Digitales y Documentos Electrónicos N° 8454, destacando el artículo 9°-

APROBACIÓN DEL COMITÉ EVALUADOR

Maestría Profesional en Ingeniería del Software con énfasis en Arquitectura y Diseño de Software

Este proyecto, titulado "**Elección de un lenguaje visual de modelado para el diseño de una arquitectura de software sobre una experiencia interactiva de GodiTours Costa Rica**", del estudiante **Juan Bautista González Núñez,** fue aprobado por el Comité Evaluador. Esta aprobación es requisito para matricular el Proyecto de Investigación Aplicada 2.

Christian Sibaja Fernández
Firmado digitalmente por Christian Sibaja Fernández
Fecha: 2023.08.18 16:38:06 -06'00'

Ing. Christian Sibaja Fernández, PH.D.
Miembro del Comité Evaluador

ALVARO CORDERO PEÑA (FIRMA)
Firmado digitalmente por ALVARO CORDERO PEÑA (FIRMA
Fecha: 2023.08.21 20:15:40 -06'0

MAP. Álvaro Cordero Peña
Miembro del Comité Evaluador

JASON ULLOA HERNANDEZ (FIRMA)
Firmado digitalmente por JASON ULLOA HERNANDEZ (FIRMA)
Fecha: 2023.08.28 10:23:32 -06'00'

Ing. Jason Ulloa Hernández, M.Sc.
Miembro del Comité Evaluador

San José, Costa Rica, 17 de agosto de 2023

Firmada digitalmente, de conformidad con la Ley de Certificados, Firmas Digitales y Documentos Electrónicos N° 8454, destacando el artículo 9°-

Tabla de Contenidos

Índice De Figuras

Índice De Tablas

Resumen Ejecutivo

GodiTours es una empresa pequeña y familia que brinda servicios de turismo natural tanto para personas nacionales como extranjeras, en un mercado creciente y de suma importancia para nuestro país, desea diferenciarse de sus competidores creando un sistema que permita una experiencia interactiva de sus clientes y la naturaleza.

El diseño de la arquitectura de software es preponderante para las organizaciones, debido a que permiten una traducción de un lenguaje de negocio a uno más técnico y viceversa, sin mencionar otras ventajas que estos ofrecen. Para la creación de un diseño se necesita un lenguaje visual que permita una compresión efectiva por parte de todos los involucrados.

Los lenguajes visuales de modelado han estado presentes desde hace décadas atrás, tratando de mejorar la comunicación de los equipos de trabajo y los involucrados; sin embargo, en los últimos años en parte, por la incursión en la industria del software de las metodologías agiles se ha ido aumentando la lista de lenguajes, lo que provoca que sea complicado para las organizaciones la elección de estos.

GodiTours al no contar con mucho recurso, desea llevar a cabo el desarrollo del prototipo para así poder entablar conversaciones con posibles inversionistas que le permitan la implementación completa de su sistema interactivo y por su falta de conocimiento requieren hacer la selección de un lenguaje que se adecue al contexto.

El contexto dentro de la ingeniería de software es un pilar muy importante para poder brindar soluciones que se adecuen a la necesidad, esto ha sido dejado de lado en muchos aspectos al intentar copiar lo que sucede en otros campos, con la creación de estándares o marcos de trabajo que en lugar de mejorar el desarrollo de software provocan personalizaciones de estos sin tomar en cuenta ciertos principios esenciales para una adecuada implementación.

Palabras Clave

Lenguajes visuales de modelado, diseño de arquitectura de software, contexto en el desarrollo de software, principios de lenguajes visuales, diagramas, arquitectura continúa, desarrollo de software, C4, metodologías agiles.

Introducción

Generalidades

Desde el inicio de la humanidad los lenguajes le han permitido comunicarse y expresarse de diferentes maneras. Uno de los sentidos importantes en la comunicación es la vista pues ayuda a comprender el mensaje que está siendo transmitido con el lenguaje no verbal. En este aspecto, el lenguaje visual tiene gran relevancia, pues abarca todo aquello que se percibe gracias a este sentido.

De acuerdo con UNIR (2022), existen 3 tipos de lenguaje visual:

- Lenguaje Visual Objetivo: se expresa información de una manera objetiva, para que no haya lugar a múltiples interpretaciones. Este tipo de lenguaje se usa en ámbitos científicos y técnicos. Ejemplo: planos arquitectónicos.
- Lenguaje Visual Artístico: transmite un mensaje estético, para que sea subjetivo al ser interpretado. Entre estos se pueden encontrar la danza, la fotografía, la pintura, entre otros.
- Lenguaje Visual Publicitario: la finalidad es vender, por lo que intenta provocar reacciones en el espectador con sus emociones.

En la Ingeniería del Software, el uso del lenguaje visual ha sido preponderante para los equipos de trabajo ya que permite transmitir información a sus miembros y además traducir requerimientos de negocio en una notación común y adecuada para que el software pueda desarrollarse. En otras palabras, se hace uso del lenguaje visual de tipo objetivo. Como en toda industria, en la del software se ha ido evolucionando este tipo de lenguaje a estándares y marcos de trabajo que faciliten la comunicación entre las personas.

UML (por sus siglas en inglés *Unified Model Language*) fue creado en 1995 y marcó un hito para este tipo de lenguajes al ser definido un estándar de consenso que personas y organizaciones fueron poniendo en uso a partir de esa época. Conforme se ha ido avanzando y madurando en la Ingeniería del Software, nuevas formas de trabajo también han aparecido, haciendo que los lenguajes de modelado también vayan evolucionando y cambiando conforme a esto. Además, han ido apareciendo nuevas herramientas. Es necesario conocer sobre estos a fin de seleccionar el que se adecue a las necesidades de las organizaciones, vía una mejor comunicación que permita tener un punto de inicio en el desarrollo de software.

Como es de conocimiento general, el turismo se ha convertido en las últimas décadas en uno de los mayores generadores de empleos e ingresos de la economía costarricense. Según la

Oficina Económica y Comercial de España en Panamá (2021), esta industria genera un aproximado del 8% (3.796,9 millones de dólares) del PIB, además de contribuir a la fuerza laboral con un total de un 28% directo e indirecto; con lo cual se convierte en uno de los pilares más importantes de la economía.

De acuerdo con el ICT (2022) en el 2022 se tuvo un ingreso total de 2 117 860 turistas por todas las vías en Costa Rica (aérea, marítima y terrestre). Estos son números alentadores y muestran una franca recuperación posterior a la pandemia que inició en el 2020.

Costa Rica es un país que posee una gran biodiversidad. Además, su posición geográfica lo hace sobresalir sobre otros destinos, por lo que ubica en el tercer lugar de Latinoamérica en el índice de competitividad de viajes y turismo, solamente por detrás de países con un mayor tamaño como lo son México y Brasil - según indica la Oficina Económica y Comercial de España en Panamá (2021).

Brindar experiencias diferentes y modernas en este sector se ha convertido en una necesidad. En un mundo globalizado las distancias cada vez son más cortas, por lo que es necesario tener servicios que se diferencien, no solamente de otros países, sino también de emprendimientos internos, los cuales ya se encuentren en el mercado local.

Antecedentes del Problema

GodiTours es un emprendimiento familiar de capital cien por ciento costarricense, enfocado en el turismo natural, que de acuerdo con CBI (2020) es todo aquel cuyas atracciones están basadas en la naturaleza del área en específico, experimentando la flora y fauna de la zona de una manera responsable con el objetivo es proteger la naturaleza y mejorar la calidad de vida de los locales.

Este emprendimiento quiere diferenciarse de la competencia. GodiTours desea que la tecnología sea uno de los pilares del camino que decidieron tomar. Por ello, desean implementar una experiencia interactiva que mezcle tanto el mundo real como el virtual, mediante la creación de una aplicación que permita estas capacidades, con ciertas características que ellos ya tienen definidas. Dado que en el mercado las aplicaciones disponibles son aquellas que permiten conocer generalidades del país y/o búsqueda de lugares o toures, no dan una experiencia adecuada al usuario final, lo que abre oportunidades en un mercado poco explotado.

Definición y Descripción del Problema

Al ser una empresa prácticamente nueva, donde los recursos de conocimiento técnico y económicos son reducidos, ellos desean asumir el desarrollo de la aplicación y pilotearla, con el fin de poder obtener mayores réditos y tomar como un punto de partida para obtener mayor capital que permita mejorar la experiencia turística, sí como cuidar los territorios en donde realizan sus actividades.

Según el ciclo de vida de desarrollo de software, después de la etapa de planeación se encuentra la fase de diseño, la cual es de vital importancia debido que es donde se analizan los requerimientos obtenidos y se identifican las posibles soluciones para poder cubrir las necesidades del cliente. En esta fase es donde, comúnmente, los equipos empiezan a realizar los diseños de la arquitectura del software que se va a desarrollar. Uno de los entregables es el diagrama de la arquitectura, que se expresa mediante un lenguaje visual que describe los elementos que van a ser parte del software por desarrollar y cómo se comunican entre ellos.

Estos diseños son relevantes debido a que permiten un entendimiento entre las partes y miembros del equipo, para poder llevar a cabo la implementación. Aunque conforme se avance en la implementación los diseños vayan cambiando (como consecuencia de decisiones que se tomen en el camino), son un punto de partida relevante pues aclaran la ruta que se debe seguir.

Como se mencionó anteriormente, existen muchos lenguajes de modelado que son usados para la creación de diagramas, por lo que a los equipos de trabajo se les dificulta la selección de un lenguaje de modelado, ya que no hay un conjunto formal de criterios. Además, se deben considerar diferentes variables para hacer la escogencia adecuada.

En la Figura 1. Diagrama Causa-Problema, se pueden observar las diferentes razones por las cuales la elección de un lenguaje visual de modelado se vuelve complejo para GodiTours. Dentro de este análisis preliminar se destacan los siguientes puntos:

- Conocimiento de los lenguajes visuales: debido a que la empresa posee recursos de un nivel principiante, el conocimiento es muy básico sobre estos.
- Recursos limitados: los recursos tanto de conocimiento como económicos son limitados en esos momentos para la organización, por lo que no se puede recurrir a consultorías externas que ayuden, no solamente en este punto, sino en todo el desarrollo.

- Información sobre cómo elegir el lenguaje visual: la organización no posee claridad sobre qué se debe tomar en cuenta para elegir un lenguaje visual de modelado sobre otro y cuál debe ser el proceso para usarlo.

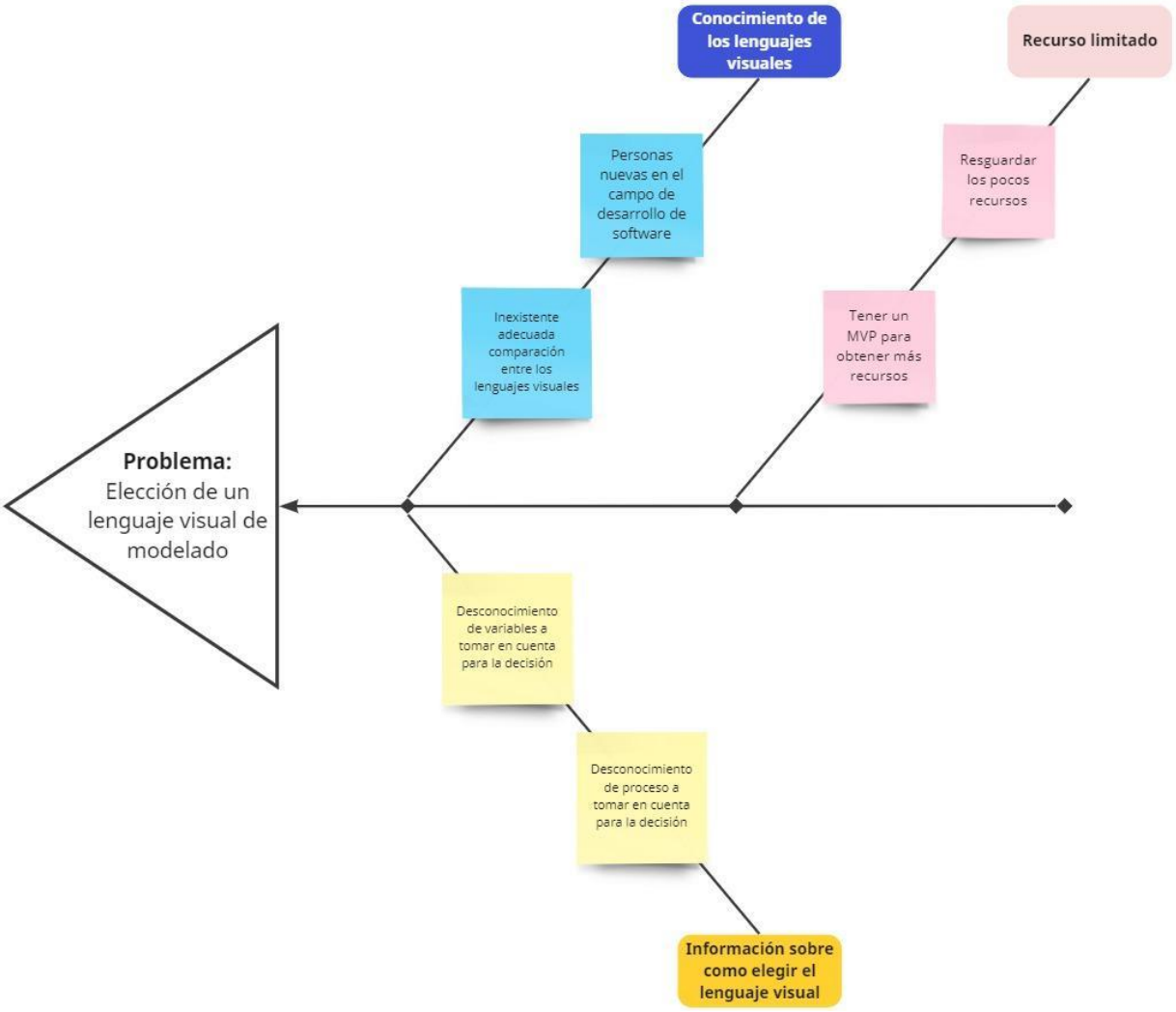


Figura 1. Diagrama Causa-Problema
Fuente: Confeccionado por el autor

GodiTours desea desarrollar dicha aplicación, pero, al tener recursos limitados, necesita una guía que les permita elegir de una manera adecuada cuál es el lenguaje visual de modelado que más se adecue a sus necesidades.

Justificación

A través de su historia, el ser humano ha utilizado diagramas, dibujos o bocetos para permitir una comunicación clara y fluida. La Ingeniería del Software no escapa de esto. Cherubini et al. (2007) detallan los diferentes escenarios en que los ingenieros usan este tipo de herramienta para facilitar o apoyarse en la comunicación. La Figura 2. *Escenarios de uso de diagramas o bocetos*, contiene la lista de aquellos puntos que son considerados relevantes para esta investigación. Es posible concluir que los diagramas son herramientas útiles para todo el ciclo de desarrollo de software.

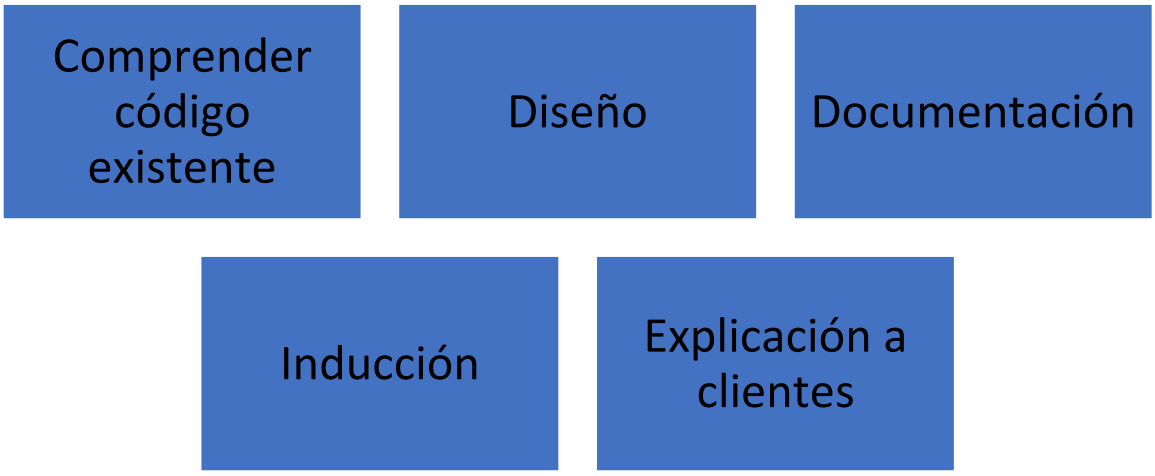


Figura 2. Escenarios de uso de diagramas o bocetos
Fuente: Confeccionada por el autor con datos de Cherubini et al. (2007)

De acuerdo con AWS (s.f) son varios los beneficios que tienen los diagramas de una arquitectura de software en la etapa de diseño, entre los que se pueden resaltar:

- Colaboración: ayuda a que el equipo de trabajo y los diferentes roles: desarrolladores, aseguradores de la calidad, arquitectos, entre otros, puedan comprender el objetivo del negocio, además, que logren acuerdo entre las partes sobre decisiones a nivel de tecnología y de procesos y dejarlas documentadas.
- Reducción de riesgos: ayuda a reducir los posibles riesgos que se pueden llegar a dar, debido a que permiten observar puntos de falla, falencias en el sistema por

desarrollar o donde se deben enfocar mayores esfuerzos de pruebas debido a las probabilidades de fallas.

- Eficiencia: al permitir una vista del sistema y su estructura los involucrados pueden encontrar fallas en el entendimiento del negocio y resolverlos de una mejor manera.
- Escalabilidad: los diagramas, al tener los diferentes flujos entre los componentes del sistema, habilitan la identificación de cuellos de botella o lugares donde se deben considerar más variables a la hora del desarrollo.

Los beneficios anteriores resaltan la importancia de que todo desarrollo de software posea un diagrama de arquitectura, como parte de su documentación, que establezca ese punto de inicio y referencia durante el diseño, la construcción y las pruebas del sistema, permitiendo mejor comunicación entre los participantes del proceso.

Contar con un lenguaje visual de modelado se vuelve importante debido a que permite tener una notación estandarizada que propicia la comunicación fluida de todo el equipo de trabajo

Para este caso en específico la escogencia de un lenguaje de modelado es un aspecto conducente a un mejor desarrollo del software y una comunicación más clara entre las partes.

Viabilidad

En la industria se sabe que los lenguajes visuales de modelado son importantes para el desarrollo de software. Sin embargo, se carece de estudios que permitan evaluarlos entre sí y poder tomar decisiones sobre su uso acorde con los diferentes contextos que poseen las organizaciones. A continuación, se describirán los puntos que hacen factible esta investigación.

Punto de Vista Técnico

Los conocimientos de la parte técnica se tienen, debido a la experiencia profesional y estudios previos que se han realizado, y a la participación que se ha tenido en diferentes proyectos de software en los cuales se han tenido diversos roles que permitieron la adquisición y puesta en práctica de los conocimientos. Además, conforme la investigación se desarrolle, el conocimiento que se empiece a obtener servirá para el cumplimiento de los objetivos propuestos y que son detallados más adelante.

Punto de Vista Operativo

Para esta investigación se ha conseguido vasta documentación, tanto antigua como moderna, proveniente de estándares o marcos creados recientemente. Se ha encontrado literatura conceptual y estudios sobre el uso de diversos lenguajes visuales de modelado que permite entender sus ventajas y desventajas, así como identificar diferentes contextos de uso de estos.

Punto de Vista Económico

La investigación se enfoca en un software en específico y en los lenguajes visuales más usados actualmente, por lo que el costo teórico del proyecto será cubierto por el autor.

De acuerdo con Glassdoor (s.f.), el salario mensual en promedio para un ingeniero de software en Costa Rica es de ₡1,705,604, lo que da un monto por hora de ₡10,152. El costo total de este proyecto se puede observar en la Tabla 1. Costo teórico de la investigación.

Costo Hora Promedio	Horas Semanales	Duración Investigación (Semanas)	Costo Total
₡10,152	30	13	**₡3,959,280**

Tabla 1. Costo teórico de la investigación
Fuente: Confeccionada por el autor con datos de Glassdoor (s.f.)

Objetivos General y Específicos

Objetivo General

- Evaluar una herramienta de selección de un lenguaje de modelado para el diseño de una arquitectura de software para una experiencia interactiva de GodiTours Costa Rica.

Objetivos Específicos

- Investigar los lenguajes de modelado mayormente utilizados en la actualidad para el establecimiento de sus ventajas y desventajas.
- Diagnosticar cuáles serían las variables para la herramienta de selección de un lenguaje de modelado de acuerdo con la industria y conocimiento de expertos.

- Diseñar la herramienta de selección de lenguajes de modelado cumpliendo con lo obtenido en el diagnóstico y de acuerdo con las prácticas actuales de la industria de la ingeniería del software.
- Aplicar la herramienta de selección de lenguajes de modelado en la organización GodiTours para el análisis de los resultados preliminares de su uso.
- Valorar críticamente el lenguaje de modelado obtenido a través de la herramienta creada.

Alcances y Limitaciones

Alcances

El alcance de esta investigación se limita a lograr la aplicación de la herramienta de selección de lenguajes de modelado para la organización GodiTours, específicamente en el software de la experiencia interactiva que ellos desean. Para esto se deben confeccionar los siguientes entregables:

- Estudio de los lenguajes visuales de modelado que son usados actualmente en la industria.
- Investigación de variables por ser utilizadas para la creación de la herramienta.
- Elaboración de la herramienta de selección de lenguajes de modelado de acuerdo con el diagnóstico obtenido y las mejores prácticas.
- Aplicación de la herramienta de selección para su valoración para así obtener los resultados y poder evaluar la factibilidad de uso de esta.

Limitaciones

Esta investigación se basa en el diseño de una herramienta para la selección de un lenguaje de modelado para el desarrollo de una experiencia interactiva específicamente de la organización. Dentro de las limitaciones se encuentran los siguientes puntos a resaltar:

- Restricción de información por parte de la organización GodiTours.
- Ausencia de conocimiento tanto técnico como administrativo por parte de la organización.
- Los requerimientos de la aplicación por desarrollar serán elaborados por la organización GodiTours.

- La aplicación de la herramienta solamente se dará para el caso en específico y su pertinente evaluación también.

Marco de Referencia

GodiTours es un emprendimiento familiar localizado en la ciudad de San Ignacio de Acosta, dedicado en su mayoría a brindar servicios de turismo de eco-aventura, dentro de los cuales se pueden destacar:

- Cañonismo
- Rápel
- Senderismo

Este emprendimiento nace debido a que ellos consideran que pueden mostrar una cara diferente de dicho cantón, mostrar sus bellezas naturales y qué mejor manera que hacerlos por las mismas personas que viven en este lugar. Como es comunicado por los propios habitantes con el eslogan "Mi pueblo promete".

A continuación, se enlista la información general de GodiTours:

- **Nombre**: GodiTours Costa Rica
- **Ubicación**: San Ignacio de Acosta
- **Año de Creación:** 2018
- **Tipo de Empresa:** Servicios Profesionales - Ecoturismo
- **Cantidad de empleados:** 5
- **Página web:**
    - https://www.facebook.com/goditours/
    - https://www.instagram.com/goditourscr/?hl=es

Estado de la cuestión

La búsqueda de información relevante para esta investigación se hizo de manera que permitiera cubrir los objetivos que se propusieron en secciones anteriores. Por ello, se plantea una búsqueda de datos relacionados con el uso de los diagramas en la Ingeniería del Software, el uso de los diferentes lenguajes visuales de modelado, junto con el contexto de su aplicación, además de sus ventajas y desventajas.

Dado que la investigación está enfocada en un tipo de software en específico y con el fin de ampliar la información obtenida, en la búsqueda también se usan criterios diferenciados, pues hay posibilidad de que esta no se encuentre completamente consolidada.

Planificación de la revisión

Esta fase se realiza con la idea de conocer y aprender sobre los diferentes aportes, no solo con fines académicos, sino también en el ámbito profesional que, aunque no tenga una estructura formalizada, permite una vista real que es necesario observar y tomar en cuenta en este tipo de investigaciones.

Formulación de la pregunta

Se limita el esfuerzo de búsqueda de acuerdo con la pregunta que se desea responder, ante esto las 3 secciones siguientes detallan cómo se realiza.

Foco de la pregunta

La respuesta a la pregunta debe ser concisa y clara, que no permita ambigüedades, ya sea de forma positiva o negativa. De acuerdo con lo detallado en Definición y Descripción del Problema, lo que se desea es formular una manera de elegir un lenguaje visual de modelado de acuerdo con las necesidades de la organización y del contexto en el cual se desarrollará el software.

Pregunta

La pregunta por resolver es la siguiente:

¿Es posible diseñar una herramienta de selección de un lenguaje de modelado para el diseño de arquitectura de software de acuerdo con las necesidades de la organización y el contexto del desarrollo?

Ejecución de la revisión

Palabras clave y sinónimos

Las palabras o frases mostradas en la Tabla 2. Listado de Palabras son aquellas que se consideran clave para la identificación general de documentos relacionados con esta investigación. En la Ingeniería del Software, la mayoría de las publicaciones más recientes se encuentran en el idioma inglés, también se detallan en ese idioma.

Como ha sido mencionado anteriormente, el despliegue de esta lista no limita que se puedan usar otras palabras que permitan complementar la búsqueda para obtener la información necesaria.

Palabra	Idioma: inglés
Lenguaje Visual de Modelado	Visual Modeling Language
Ingeniería del Software	Software Engineering
Diagrama	Diagram
Boceto	Sketch
Arquitectura de Software	Software Architecture
Notación Gráfica	Graphical Notation
Importancia	Importance
Uso	Use

Tabla 2. Listado de Palabras
Fuente: Confeccionada por el autor

Criterios de inclusión y exclusión

Inclusión	Exclusión
Trabajos que contengan investigaciones sobre el uso de los lenguajes visuales de modelado en proyectos de software	Investigaciones donde en su título no contenga relación con implementación o comparación de los lenguajes visuales de modelado
Trabajos que comparen el uso de los lenguajes visuales de modelado en proyectos de software	Investigaciones donde a pesar de que contiene información relevante, se encuentra con otro objetivo en su investigación final
Fuentes confiables que realicen estudios sobre los lenguajes visuales de modelado	Investigaciones cuyo campo de investigación no sea la Ingeniería del Software
Fuentes que, por su título, llamen la atención del lector	

Tabla 3. Criterios de inclusión y exclusión de fuentes
Fuente: Confeccionada por el autor

Identificación de Fuentes

Las fuentes que se van a tomar en cuenta son todas aquellas que cuentan con un respaldo, tanto académico como profesional, en el área de la Ingeniería del Software, haciendo uso de los dos idiomas, con el fin de extender la información que se puede obtener. Además, se priorizarán aquellas fuentes que sean de fácil acceso, esto quiere decir que se puedan obtener los permisos para acceder a ellas.

Trabajos de investigación a utilizar

Título	Autor(es)	Año	Palabras Claves	Enlace

Modeling Software Development Methodologies: A Conceptual Foundation	Cesar Gonzalez-Perez Brian Henderson-Sellers	2007	Modeling, Software Development Methodologies	https://www.sciencedirect.com/science/article/abs/pii/S0164121207000696
A Survey of Modeling Language Specification Techniques	Dominik Bork Dimitris Karagiannis Benedik Pittl	2019	Modeling Language Specification	https://www.sciencedirect.com/science/article/abs/pii/S0306437919303035?via%3Dihub
Sketches and Diagrams in Practice	Sebastian Baltes Stephan Diehl	2014	Sketches Diagrams	https://dl.acm.org/doi/10.1145/2635868.2635891
Modeling Languages in Industry 4,0: An Extended Systematic Mapping Study	Andreas Wormann Olivier Barais Benoit Combemale Manuel Wimmer	2019	Modeling Languages	https://dl.acm.org/doi/10.1007/s10270-019-00757-6

Tabla 4.Trabajos de investigación a utilizar
Fuente: Confeccionada por el autor

Resumen de los resultados

Nombre	**Modelling Software Development Methodologies: A Conceptual Foundation**
Objetivos de la investigación	1-Explicar los modelos estructurales. 2- Validar cómo los modelos estructurales pueden describir metodologías de desarrollo de software.
Resumen	Los autores expresan la definición sobre lenguajes de modelado que se basa en una abstracción del mundo real con el fin de comprender el funcionamiento de este. También describen cómo estos pueden tener diferentes niveles de representación. Además, definen tres características: son organizados, no están compuestos por palabras necesariamente y la notación no es parte de este.
Resultados Obtenidos	1- Los modelos siempre son homomórfico, esto indica que los diseños pueden interpretarse en la realidad y en el papel. 2- Algunos modelos actuales (UML, entre otros) tienen deficiencias para representaciones reales.

Principales Conclusiones	1- Los modelos usados como ejemplo en esta investigación tienen faltantes para poder representar ciertos escenarios del mundo real lo que impide su uso adecuado. 2- Notación gráfica se separa totalmente de los modelos, hay lenguajes que no poseen metamodelos (refieren al estándar ISO/IEC 24744).

Tabla 5. Resumen investigación 1
Fuente: Confeccionada por el autor

Nombre	**A Survey of Modeling Language Specification Techniques**
Objetivos de la investigación	1- Identificar técnicas de especificación para todos los aspectos conceptuales de un lenguaje de modelado.
Resumen	Los autores definen diversos conceptos con el fin de hacer una evaluación entre diferentes lenguajes de modelado, dentro de las cuales es relevante mencionar: conectores, particiones de diagramas, notación e interoperabilidad.
Resultados Obtenidos	1-Heterogenidad entre las especificaciones de los lenguajes 2-Los lenguajes estudiados tienen bastantes limitaciones que complican su uso. 3- No hay referencias en los lenguajes hacia meta-metamodelos lo que provoca dificultad de uso.
Principales Conclusiones	1-La importancia de los lenguajes visuales de modelado es real, debido a su vital papel en el desarrollo de software y conversaciones que se dan. 2- Los lenguajes estudiados presentan pequeñas similitudes con respecto de los conceptos de un metamodelo.

Tabla 6.Resumen investigación 2
Fuente: Confeccionada por el autor

Nombre	**Sketches and Diagrams in Practice**
Objetivos de la investigación	1- Investigar el uso de diagramas y bocetos en la Ingeniería del Software.
Resumen	Los autores investigan cuál es el uso que se le da a los diferentes diagramas y qué tipo usan. A pesar de que UML es el lenguaje más usado se sigue usando solamente la notación de una manera informal. La mayoría de los diagramas son desechados sin uso alguno para un futuro.
Resultados Obtenidos	1- Los bocetos o diagramas son usados en la industria para diferentes tipos de conversaciones. 2- Aunque UML es usado mayoritariamente de una manera informal, la notación usada es muy variable en cada empresa.

Principales Conclusiones	1- Los diagramas, aunque sean informales, son relevantes en muchos aspectos del flujo de trabajo. 2- El entendimiento entre las partes es uno de los motivos principales por los que se hace uso de estos. 3- Como otros tipos de documentación, los diagramas no están exentos de malas prácticas y pueden quedar desactualizados en algún punto del desarrollo.

Tabla 7.Resumen investigación 3
Fuente: Confeccionada por el autor

Nombre	**Modeling Languages in Industry 4.0: An Extended Systematic Mapping Study**
Objetivos de la investigación	1- Evaluar el uso de modelado en la industria 4.0 a través de los lenguajes de modelado.
Resumen	Los autores realizan un estudio para conocer el uso sobre los lenguajes de modelado en la nueva industria. Ante esto, hacen un estudio de fuentes. Encuentran que UML y BPMN aparecen como las predominantes; sin embargo, eso no es la norma. Se nota que hay faltante de estudios en muchas áreas de estas, y que la mayoría de los trabajos están enfocados en las bases de los lenguajes.
Resultados Obtenidos	1- Aplicar estos lenguajes de cierto modo se ha quedado corto en la industria. 2- La disciplina aún está en proceso de maduración debido a que se encuentran más publicaciones en las bases de estos.
Principales Conclusiones	1-Se necesita realizar más estudios evaluativos en la industria para obtener mayores resultados de su aplicación. 2- Los lenguajes y las representaciones de conocimiento no están siendo combinadas, con lo cual se espera que futuras investigaciones realicen esto, debido a que ya se están iniciando algunas.

Tabla 8. Resumen investigación 4
Fuente: Confeccionada por el autor

Conclusiones de los resultados

Dentro de los trabajos seleccionados se logra observar cómo en la industria se han investigado las bases, las aplicaciones y se ha realizado una comparación teórica entre los diferentes lenguajes de modelado. A pesar de llevar años en la industria de la Ingeniería del Software estos no son usados y aprovechados como se debería. Sin embargo, se observa la relevancia que estos tienen cuando se usan, pues permiten un mejor flujo de trabajo, así como mejores conversaciones.

Se puede observar que, sí hay un faltante, a nivel comparativo, del uso de los diferentes lenguajes que existen, debido a que la información que se encuentra solamente se basa en teoría, lo que imposibilita tener escenarios para conocer cuáles son los beneficios y perjuicios reales a la hora de seleccionar un lenguaje sobre otro. Por lo anterior, la presente investigación puede aportar luces sobre este tema, al desarrollar una herramienta que ayude a la toma de decisiones de acuerdo con características establecidas.

Marco Conceptual

"*Una arquitectura es una organización fundamental del sistema entre sus componentes, la relaciones de estos y también del entorno donde se encuentran, además de aquellos principios que guían su diseño y evolución*": esta es la definición dada en el estándar de la IEEE 1471. Con referencia a lo anterior, dentro de la arquitectura propiamente, el diseño es una parte relevante de esta, lo cual ofrece múltiples beneficios.

De acuerdo con la anterior definición, Sommerville (2015) conceptualiza que un diseño de arquitectura es un proceso que identifica los componentes de este, cómo se comunican entre ellos y cómo se controla el sistema. Además, la arquitectura es el enlace entre la especificación y el diseño del software. Asimismo, expone que es un proceso creativo como tal y depende también del contexto y entorno donde se desarrolle el software. Se logra señalar la relevancia que tiene esta parte del desarrollo de software para una implementación adecuada de este.

Los beneficios que se obtienen al tener una arquitectura para el equipo de trabajo son bastantes, dentro de los que se pueden resaltar, de acuerdo con Tsenov (2022) y Sommerville (2015), los siguientes:

- Base sólida: los miembros del equipo logran llegar tener un lenguaje común para la implementación del software.
- Comunicación con interesados del proyecto: permite tener una comunicación fluida con las personas externas al equipo debido a que la compresión del software se vuelve sencilla con los diagramas y diseños realizados.
- Reutilización: dependiendo del contexto, partes del diseño se pueden reutilizar en futuros proyectos.

- Análisis del sistema: permite realizar análisis complejos para los requerimientos no funcionales del sistema, además se pueden observar puntos de mejora desde antes de la implementación.

Como resultado de los puntos mencionados anteriormente, y lo expuesto en la justificación de este proyecto, expresar el diseño de una arquitectura con un diagrama adecuado contribuye a que las probabilidades de éxito de un proyecto aumenten.

Kruchten (1995) presentó un modelo de arquitectura de software que se hizo universal en la industria poco después publicación, pues dio una perspectiva holística a los diseños que se realizaban. El modelo, llamado **4+1**, engloba los diferentes puntos de vista y diseños que se deben realizar para una arquitectura, con el fin de cubrir las aristas que componen el software y, además, permitir una conversación fluida entre los involucrados.

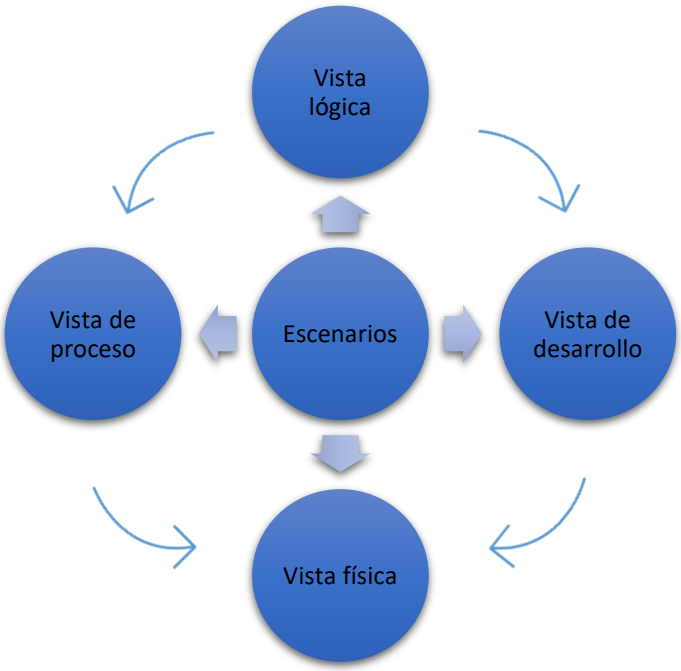


Figura 3. Modelo de Arquitectura 4 + 1
Fuente: Confeccionado por el autor con datos de Krutchen (1995)

En la Figura 3. Modelo de Arquitectura 4 + 1, se detalla cómo se relacionan las vistas entre sí. A continuación, se describe cada una de ellas:

- **Vista lógica:** diseño o diagrama que provee una vista en términos de servicios para los usuarios. El sistema se separa en forma de objetos de acuerdo con el dominio del problema.

Esta vista, se enfoca en describir la funcionalidad del sistema desde la perspectiva del usuario final.

- **Vista de proceso:** permite enfocarse en los requerimientos no funcionales como, por ejemplo, el rendimiento del software. Un proceso es una agrupación de tareas que forman una unidad.
- **Vista física:** todo software se ejecuta en redes de computadoras o nodos, por lo cual esta vista detalla todo lo necesario para que el software se logre ejecutar de una manera adecuada.
- **Vista de desarrollo:** se enfoca en la organización del software en el ambiente de desarrollo y cómo los componentes se comunican entre sí.

Cabe destacar que Kruchten (1995) menciona que no es obligatorio crear todos los diagramas, que estos deben ser creados de acuerdo con el dominio y contexto de software, semejante a lo mencionado en párrafos anteriores. Asimismo, para crear los diagramas, Kruchten acota que la notación usada puede variar de acuerdo con el equipo de trabajo y que estos pueden hacer uso de diferentes herramientas, por lo que toman preponderancia los lenguajes visuales de modelado en esta etapa.

Un lenguaje visual de modelado es aquel que permite una comunicación visual sobre las diferentes vistas que posee un software, permitiendo un diseño de arquitectura adecuado para que los equipos de trabajo logren una comunicación eficiente y efectiva junto con los involucrados del proyecto.

En los últimos años, conforme la Ingeniería del Software ha ido madurando, este tipo de lenguajes han ido también creciendo en cantidad debido a la necesidad, como mencionan Woods y Hilliard (2005), UML a pesar de no ser considerado como una herramienta para este contexto, es el más utilizado junto a cajas y líneas para representar las arquitecturas. Además, mencionan cómo se ha tenido que usar, junto al modelo 4 + 1, con el fin de tener una mejor representación en los diseños.

Con el fin de tener un entendimiento más profundo sobre los diferentes lenguajes, se han seleccionado algunos de ellos para realizar un estudio que permita entender cuáles son las características, así como ventajas y desventajas, que cada uno de ellos posee, a fin de tener una comparativa entre ellos que ayude a los objetivos de esta investigación. La selección de estos

lenguajes se hizo con base en la experiencia, pues no fue posible encontrar publicaciones que con datos sobre los más usados en diferentes proyectos de software.

UML

UML nació en 1995 a partir de la propuesta base de Booch, Jacobson y Rumbaugh (entonces en Rational Software), como un lenguaje visual que permite documentar los artefactos de un sistema de software, incluyendo desde procesos de negocios hasta niveles más detallados - como las declaraciones de los lenguajes de programación. Después de un proceso de consenso, en 1997, el Object Management Group lo adoptó y promovió como estándar. Su uso se ha extendido desde entonces y se definió como estándar internacional vía ISO/IEC 19501:2005.

Este lenguaje provee una notación estandarizada. Cada elemento del diagrama tiene un concepto asociado, por lo cual no importa cuál sea el proyecto de software el diagrama se podrá dar una interpretación unívoca a cada elemento del modelo.

De acuerdo con Ave Coders (2021) los diagramas se pueden dividir en dos:

- Diagramas Estructurales: diagramas que representan la estructura propia del software en sus diferentes modos. Entre estos, podemos mencionar: clases, componente, objeto.
- Diagramas de Comportamiento: aquellos que representan cómo se comportan las diferentes entidades entre sí, permitiendo describir el proceso de acuerdo con los pasos que tienen que llevarse a cabo de acuerdo con el objetivo deseado. En este se encuentran: caso de uso, máquina de estado, interacción.

La idea detrás de que se tengan diversos diagramas, un total de 13, es poder conseguir una vista completa del software y que los diferentes roles (en equipos de trabajo) puedan cubrir sus necesidades a la hora de comprender y explicar la solución a los problemas que se busca resolver.

En la Figura 4. *Ejemplo Diagrama de Clases UML*, se puede observar un diagrama de clases UML, para una tienda en línea. Contiene las clases que van a ser usadas, las variables y relaciones entre estas.

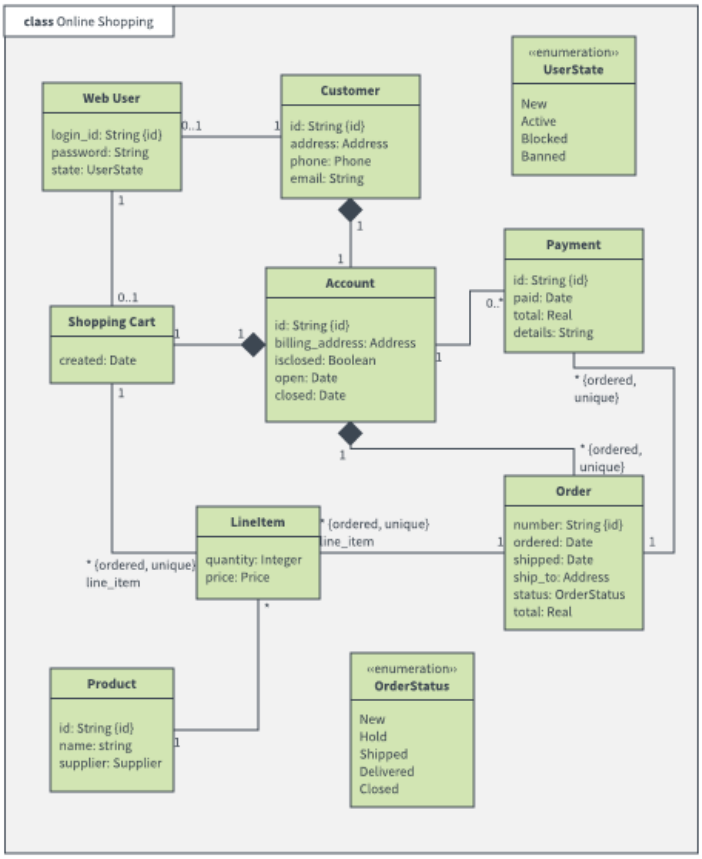


Figura 4. Ejemplo Diagrama UML
Fuente: Lucidchart (2019)

Ventajas

- Legibilidad: si se llega a usar de la manera adecuada, leer los diferentes diagramas es sencillo debido a que se puede comprender fácilmente cuál es el objetivo debido al significado y orden que cada objeto posee.
- Comunicación entre roles: al tener diversidad de diagramas, permite que los diferentes roles puedan entablar las discusiones adecuadas para comprender el contexto de un problema o las características de diversas opciones de solución.
- Estandarización: al ser un estándar, la comunicación entre organizaciones o equipos se vuelven eficaces debido a que el entendimiento es mejor entre las partes.
- Generación de código: en la industria hay herramientas que pueden generar, parcial o totalmente, código de programación o descripciones necesarias para implementar el software modelado por los diagramas.

Desventajas

- Consumo de tiempo: al tener una notación estandarizada, el equipo de trabajo debe de seguir las reglas, además de la cantidad de diagramas que se pueden realizar el planeamiento puede llevar más tiempo de lo pensado. Esto puede provocar retrasos al implementar el software debido a que esta fase solamente vamos a tener *expresiones de diseños*.
- Complejidad: si los sistemas son complejos, la representación y creación de diagramas se pueden volver un dolor de cabeza para los equipos de trabajo.
- Curva de aprendizaje: a pesar de que es conocido en la industria, son pocas las personas que llegan a aprenderlo a fondo, lo que provoca que no se use todo su potencial y además no se llegue a comprender todos los diagramas.
- Costo: el costo para tener herramientas que permitan la generación de código es bastante oneroso, algo por considerar si se desea dar este tipo de utilización al lenguaje.

C4

Este lenguaje ha tomado bastante auge desde que las prácticas ágiles llegaron a la industria del desarrollo de software. Su creador, Simon Brown (2019), se inspiró en el modelo 4+1 y también UML para su creación. C4 se conceptualiza como un conjunto de abstracciones y diagramas jerárquicos, en los cuales la notación y herramientas a usar son independientes.

El objetivo principal de este lenguaje es que el proceso de diseñar sea lo más orgánico posible con el fin de enfocarse en lo que realmente importa que es la solución, permitiendo conversaciones y comunicaciones precisas entre las partes. Su nombre viene de los cuatro niveles que componen las vistas de este lenguaje:

- **C**ontexto del Sistema: es la vista del sistema desde el punto más alto. Contiene el sistema a desarrollar junto con los usuarios y otros sistemas con los cuales interactúa.
- **C**ontenedor: este contiene a un alto nivel la arquitectura propia del software a desarrollar. Se indican decisiones técnicas y cómo cada contenedor se relaciona con otros.
- **C**omponente: despliega cuáles son los componentes de cada contenedor y cuáles son sus responsabilidades dentro del contenedor.

- **C**ódigo: para cada componente se despliega cómo va a ser implementado a nivel de código. Este es un nivel opcional y solamente se recomienda utilizarlo si el componente es complejo.

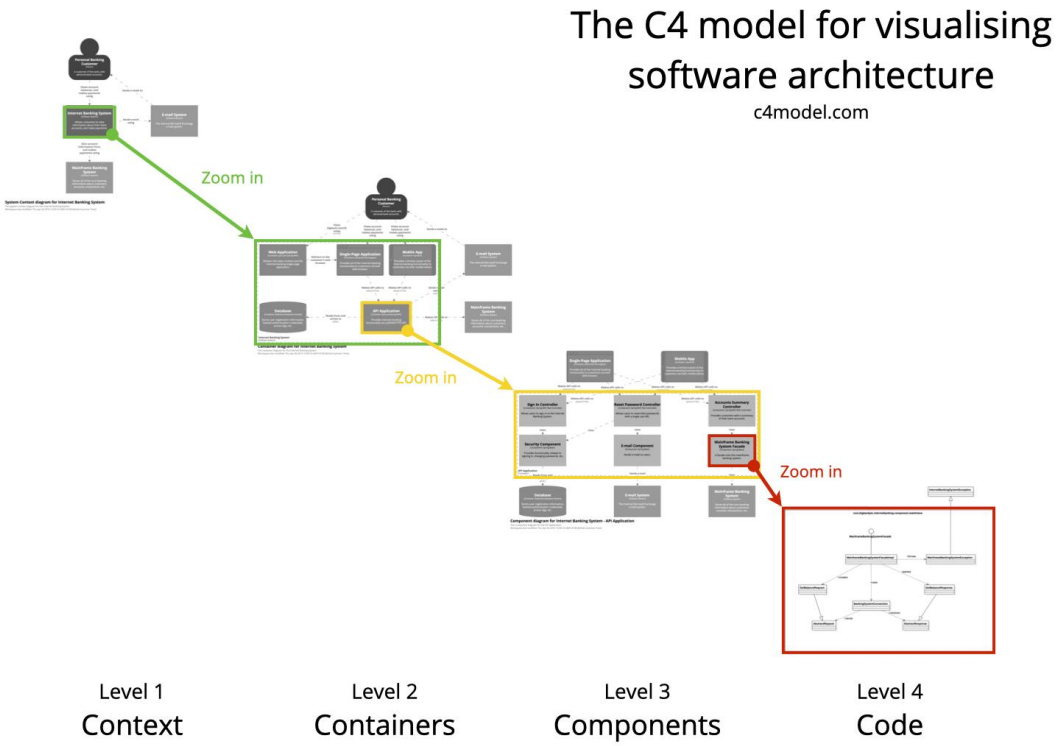


Figura 5 Ejemplo Diagrama C4
Fuente: Brown (s. f.)

La idea de la composición de C4 es ir acercándonos a vistas más detalladas cada vez mayores, esto con el fin de usar cada diagrama para diferentes perspectivas y conversaciones, como se puede apreciar en la Figura 5. *Ejemplo Diagrama C4*. Por ejemplo, la idea de los dos primeros niveles da perspectiva a nivel del negocio y cómo serían las relaciones entre usuarios y los diferentes sistemas, en cambio los dos últimos niveles son más técnicos.

Ventajas

- Estructura: al ser vistas de diferentes niveles, permite entender los niveles adecuadamente y la funcionalidad de cada parte del sistema.

- Curva de Aprendizaje: la curva del aprendizaje es mínima, debido a que la notación por utilizar se ajusta al equipo de trabajo. Al elaborar la documentación es necesario considerar las buenas prácticas.
- Flexibilidad: las diferentes formas de notación por usar permiten que se adecue el lenguaje a las necesidades del equipo de trabajo.

Desventajas

- Estandarización: al ser la notación diferente para cada equipo de trabajo, se debe tener guías o ayudas que permitan entender a personas que no estuvieron en los acuerdos tomados para la notación.
- Dificultad en organizaciones grandes: debido a su simplicidad, C4 actualmente es usado para partes específicas de software, o sistemas de software que no son grandes. La representación de sistemas de mayor tamaño se torna compleja, por lo que se requiere cierta experiencia para hacerlo apropiadamente.
- Adopción: al ser un lenguaje moderno, se ha adoptado rápidamente debido su buen acople con los métodos ágiles. Aún falta madurar su uso, en más casos reales y conseguir mayor soporte de la comunidad.

ArchiMate

The Open Group (s.f.), creadores del lenguaje ArchiMate, lo definen como un marco de trabajo desarrollado para comunicar, analizar y describir los diferentes puntos de una arquitectura empresarial. ArchiMate consta de tres capas: negocio, aplicación y tecnología; en esta última también se encuentra integrada la parte física.

Este lenguaje está en su tercera versión, gracias a actualizaciones realizadas con el fin de mantenerse al día con los cambios en el entorno empresarial y las necesidades de los usuarios de ArchiMate. La actualización más reciente data del 2022 (versión 3.2).

La Figura 6. *ArchiMate Estructura Marco de Trabajo* muestra la composición de ArchiMate. A continuación, se da una breve explicación de los aspectos y capas que lo conforman:

- Aspectos:
    - Estructura Activa: representa el *quién* (los actores). Son aquellos elementos que realizan las acciones para que se dé el comportamiento.

    - Comportamiento: aquellos procesos o eventos que se dan debido a las acciones que hacen los actores. Representa el *cómo*.
    - Estructura Pasiva: son los elementos sobre los cuales se ejecutan los comportamientos. Representa el *qué*.
- Motivación: aspecto que fue agregado en la tercera versión, con el fin de tener una parte del diagrama que muestre cuáles son las razones en las que se basa el diseño creado.
- Capas:
    - Estrategia: expresa cómo la organización desea crear valor, las capacidades del software y los recursos. Todo esto en el alcance del diseño arquitectónico.
    - Negocio: representa el modelo operacional de la organización. Es independiente de la tecnología que se vaya a usar.
    - Aplicación: describe el comportamiento e interacciones entre las aplicaciones que posee la organización.
    - Tecnológica: se refiere al comportamiento de la infraestructura tecnológica de la empresa.
    - Implementación y Migración: se enfoca en el área de aquellos elementos que soportan tanto la migración como la implementación de las arquitecturas que puede poseer una organización.

Al igual que UML, la notación de ArchiMate está totalmente definida, lo cual habilita la interpretación unívoca de los diagramas expresados con ella.

Ventajas

- Comprensión: al ser un lenguaje estandarizado, es fácil comprender si se tiene el conocimiento sobre la notación.
- Integración: el uso de los diferentes aspectos y capas permite que se tenga una vista completa de un sistema de software en el contexto de una organización.
- Estructura: el lenguaje está estructurado conforme a diversas vistas. La notación es adecuada para lograr el entendimiento de los involucrados.

Desventajas

- Complejidad: cada versión que ha tenido ArchiMate ha agregado más capas o aspectos, así elementos notacionales. Esto dificulta su uso si no se tiene un conocimiento profundo del marco de trabajo.
- Curva de Aprendizaje: dada su complejidad, la curva de aprendizaje es alta, similar a la de UML. ArchiMate se enseña poco en la Educación Superior y se usa menos que UML, por lo que su uso es menos universal.
- Adopción: de acuerdo con Victor (2023) el uso mayoritario de este lenguaje está enfocado en Europa, lo que hace que la comunidad y el soporte sea más reducido.

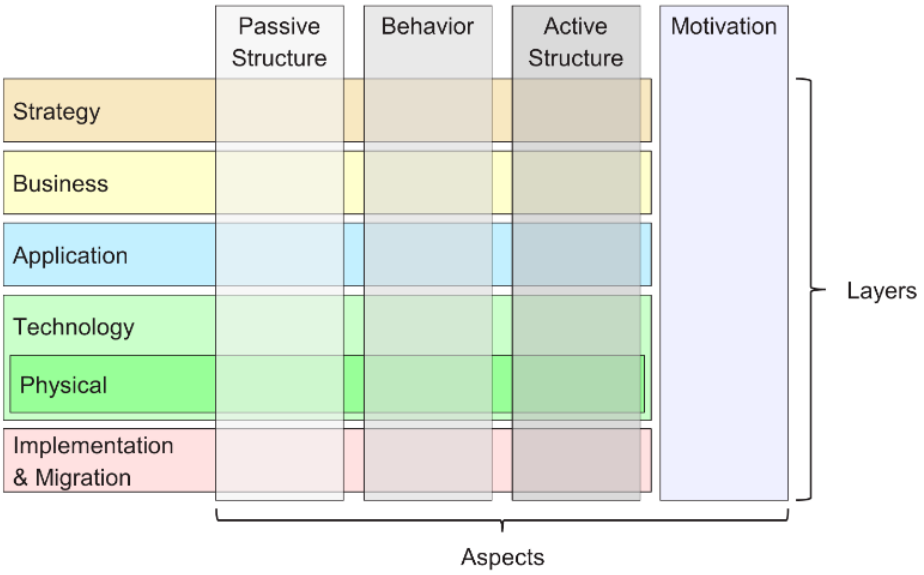


Figura 6. ArchiMate Estructura Marco de Trabajo
Fuente: The Open Group (s. f.)

BPMN

Business Process Model and Notation es un estándar que tiene como fin dotar de una notación entendible por todos los usuarios del negocio; entre los cuales se pueden mencionar: analistas, desarrolladores, entre otros. Fue creado por el Object Management Group (OMG). Su segunda versión (actual) fue lanzada en el 2011 (The Object Management Group, 2011),

Está compuesto por tres modelos básicos, los cuales se explican brevemente a continuación:

- Procesos: aquellos procesos internos de la organización, conocidos en la industria como flujos de trabajo u orquestación de servicios. Este a su vez se encuentra dividido en privado no ejecutable, privado ejecutable y público.
- Coreografías: es la definición del comportamiento esperado entre los participantes, contiene las diferentes interacciones entre estos, creando un flujo entre las partes.
- Colaboraciones: representa las interacciones entre dos o más entidades del negocio.

Además de tener una notación con respecto de los símbolos por usar, se detalla también cómo se debe de usar el texto, colores y líneas dentro de los diferentes diagramas. En la Figura 7. *BPMN Ejemplo de Proceso*, se puede observar un diagrama que posee un proceso u orquestación.

A pesar de que su creación ha sido para procesos, hay organizaciones que hacen uso de este para sus diseños de arquitectura del software debido a que permite que los involucrados tengan un mismo lenguaje que exprese el contexto de aplicaciones empresariales o componentes de software que son orquestados desde un proceso de negocios.

Ventajas

- Audiencia: debido a enfocarse en procesos permite que los diferentes roles puedan establecer discusiones para brindar soluciones o confirmar supuestos debido a su notación.
- Estándar: al crear un estándar en su notación, permite que el entendimiento entre las partes sea el mismo, y se eviten confusiones innecesarias.
- Modelos: los 3 modelos mencionados anteriormente permiten que el equipo de trabajo seleccione aquellos que más se adecuen a sus necesidades.

Desventajas

- Curva de Aprendizaje: al igual que lenguajes anteriores se deben entender completamente los aspectos generales y específicos del lenguaje, esto con el fin de hacer un buen uso de este. El documento oficial cuenta con un aproximado de 500 páginas.
- Diferencias en implementación: a pesar de ser un estándar muchos proveedores y software lo implementan con diferencias que complican la adopción de este. (Victor, 2023)
- Complejidad de representación: debido al uso de los denominados carriles de natación, se vuelve complejo representar diferentes roles y actividades.

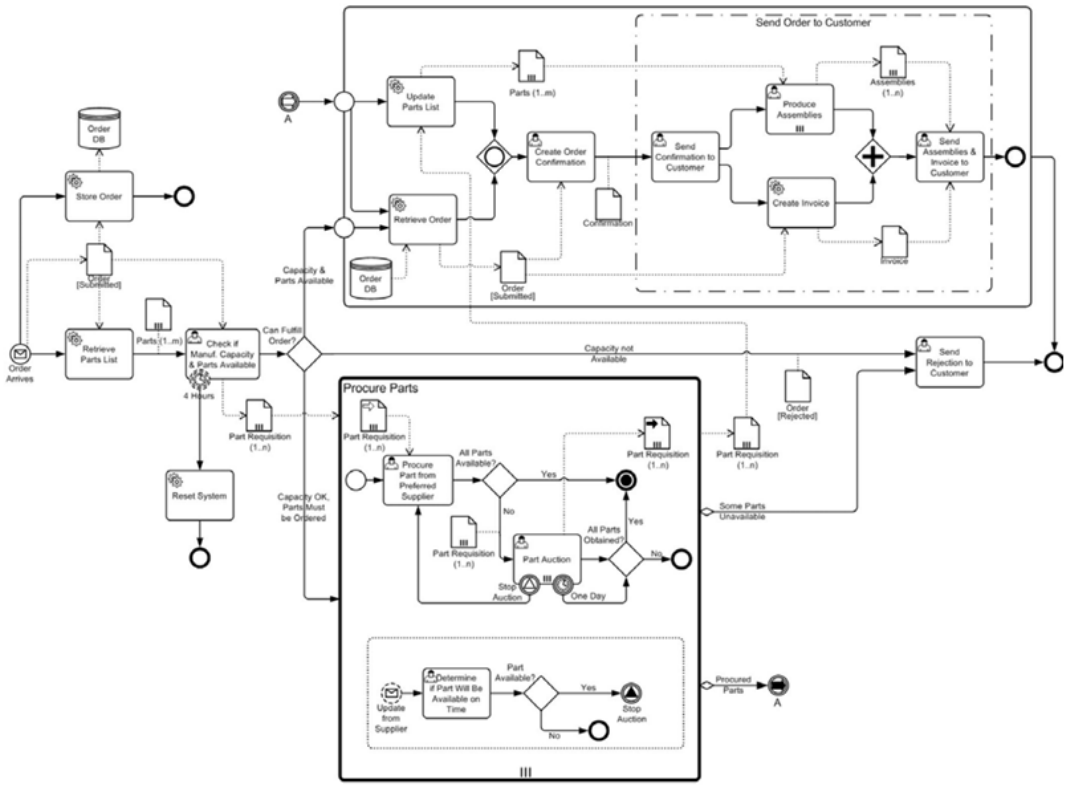


Figura 7. BPMN Ejemplo de Proceso
Fuente: The Object Management Group (2011)

Como se pudo notar, cada lenguaje tiene sus ventajas al ser usado, las desventajas pueden ser minimizadas de acuerdo con el contexto en que se encuentra el proyecto por realizar. En la Tabla 3. *Comparación de lenguajes seleccionados* se logra observar, con las variables más relevantes de la investigación, cuáles son las diferencias entre los lenguajes de modelado.

Lenguaje	Curva de Aprendizaje	Notación	Complejidad	Adopción	Generación de Código	Documentación
UML	Alta	Estándar	Media-Alta	Alta	Sí	Alta
C4	Baja	No	Baja	Baja	No	Media
ArchiMate	Alta	Estándar	Alta	Media	No	Media
BPMN	Alta	Estándar	Alta	Media	Sí	Alta

Tabla 9. Comparación de lenguajes seleccionados
Fuente: Confeccionada por el autor

Marco Metodológico

Tipo de Investigación

Debido a la naturaleza de esta investigación que persigue brindar una solución a una necesidad de GodiTours, de acuerdo con los conocimientos ya existentes, se realiza una investigación aplicada - participativa.

Alcance Investigativo

El alcance para esta investigación es exploratorio, debido a que en la industria no se encuentra cuál lenguaje visual de modelado se debe de usar de acuerdo con el contexto, a pesar de que existe mucha información. No se ha encontrado información que ayude a tomar la decisión con base en criterios.

Enfoque

El enfoque utilizado es cualitativo, pues nos enfocaremos en obtener datos de una manera abierta, en busca de conocer criterios de personas especializadas en el campo, además de acudir a la literatura sobre el tema. A esto se suma la experiencia del autor, obtenida a partir de su trabajo en diversos proyectos.

Diseño

En la Figura 8. *Diseño de la investigación* se puede apreciar la organización de las fases que esta contiene. A continuación, se detalla cada una de las fases:

- Estudio de la información disponible: se recaba información relevante para la investigación, con el objetivo de tener una base conceptual sólida.
- Creación del cuestionario: de acuerdo con la base conceptual, se elabora un cuestionario que permita recabar conocimiento experto con personas especializadas.
- Selección de sujetos: se eligen personas conocedoras y especializadas con las cuales se puedan tener puntos de vista pertinentes al seleccionar lenguajes visuales de modelado.
- Aplicación de cuestionario: se aplica el cuestionario con la colaboración de los sujetos.
- Resultados del cuestionario: al obtener los cuestionarios llenos, se agrupan los resultados con el fin de obtener patrones para realizar un análisis adecuado.
- Análisis de resultados: de acuerdo con los resultados obtenidos, se elabora una lista de perspectivas que permitan plantear la propuesta.

- Desarrollo de propuesta: en concordancia con el contexto estudiado y la organización, se elabora la herramienta de selección de lenguaje que permita hacer uso de esta para su evaluación.

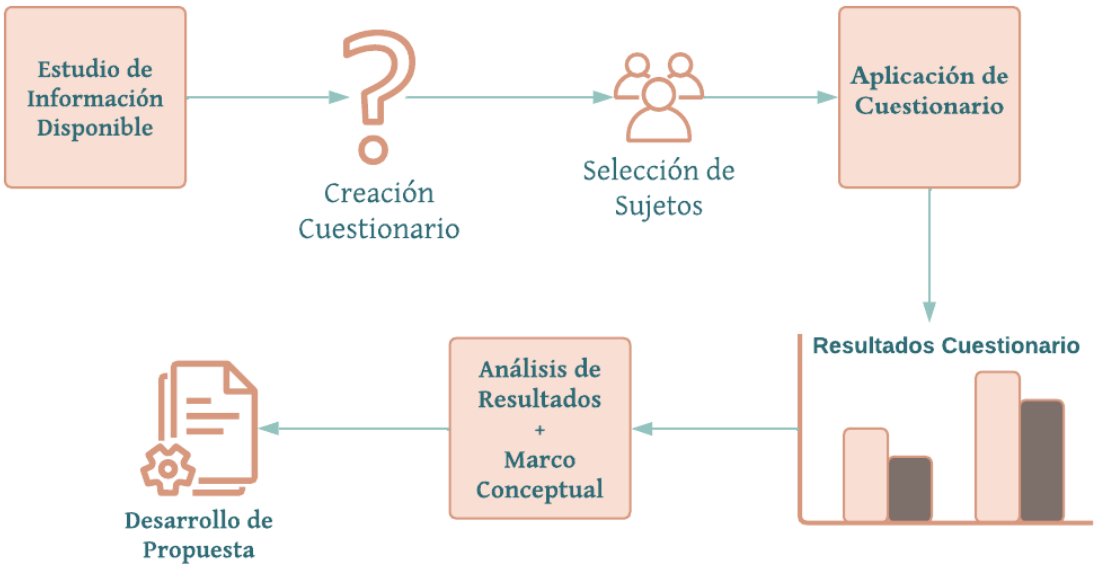


Figura 8. Diseño de la investigación
Fuente: Confeccionada por el autor

Población y Muestreo

Para esta investigación se realizan entrevistas orientadas a cumplir con los objetivos que se propusieron en el primer capítulo de esta investigación. A continuación, se presenta qué tipo de población va a ser objeto de dichos cuestionarios:

Población (roles)	**Justificación**
Desarrolladores de Software con mínimo 3 años de experiencia en diversos proyectos **Cantidad: 2 Personas**	Entender el contexto a nivel del desarrollador sobre los lenguajes y la experiencia que han tenido haciendo uso de estos.
Arquitectos de Software con mínimo 5 años de	Analizar la escogencia de los lenguajes visuales de

experiencia en diferentes proyectos **Cantidad: 2 Personas**	modelado de acuerdo con la experiencia real y cuáles fueron los criterios usados.
Administradores de Productos con mínimo 5 años de experiencia en diferentes proyectos **Cantidad: 2 Personas**	Obtener información sobre el uso que se ha dado a los lenguajes visuales de modelado a nivel de negocio y comprender su uso desde una perspectiva de la Ingeniería del Software.

Tabla 10. Muestra de la población
Fuente: Confeccionada por el autor

Análisis de la situación

El análisis realizado se divide en dos partes:

1. **Aplicación de cuestionario a expertos**: se desarrollan entrevistas estructuradas con la población seleccionada, a fin de evaluar el proceso en la toma de decisiones y el uso que se les da a los lenguajes visuales de modelado según la perspectiva de cada rol seleccionado. Las entrevistas se estructuran con base en el cuestionario diseñado previamente. Ver detalle de la entrevista en el Apéndice 1, *Cuestionario a expertos*.
2. **Estudio del contexto de GodiTours:** se realizan entrevistas con personas de la organización a fin de entender el objetivo del software que se desea desarrollar y entender su contexto. Ver detalle de la entrevista en el Apéndice 2, *Cuestionario GodiTours*.

La separación del análisis busca entender la perspectiva de cada parte involucrada en esta investigación. Con esta información, se puede obtener mejor criterio para realizar una propuesta de solución pertinente al problema.

Análisis de las entrevistas

Aplicación de cuestionario a expertos

Este análisis se divide de acuerdo con la población y el muestreo. El detalle de las entrevistas puede verse en el Apéndice 3, *Información cuestionario a expertos*.

Desarrolladores de Software con mínimo 3 años

Luego de obtener la información de parte de los desarrolladores, estos fueron los resultados para esta investigación:

- Conocimiento de lenguajes visuales de modelado: UML es el lenguaje conocido. Sin embargo, se menciona que el uso de este en el trabajo no es tan relevante, pues los bocetos toman mayor preponderancia a la hora de realizar los diseños.
- Toma de decisión: el contexto del software es lo relevante para decidir sobre cuál lenguaje elegir, en las organizaciones el hecho de que se seleccione uno como el estándar está alejado de la realidad debido a la independencia y la organización de equipos.
- Característica relevante: el aprendizaje y la flexibilidad del lenguaje son las cualidades mencionadas, pues se requiere que la información por transmitir sea eficiente y efectiva.
- Característica no relevante: estandarizado, debido a la naturaleza de la organización y metodología de desarrollo usada. No interesa si el lenguaje es estandarizado debido a que lo relevante es que el mensaje sea transmitido adecuadamente.
- Uso de los diagramas: los diagramas dentro del equipo no son usados comúnmente. Sin embargo, se tiene conocimiento que para ciertos involucrados ofrece algunas ventajas para el entendimiento de flujos y dependencias.

Arquitectos de Software con mínimo 5 años

Los siguientes puntos son los más relevantes de las conversaciones con los arquitectos de software entrevistados:

- Conocimiento de lenguajes visuales de modelado: UML y C4 son los lenguajes mencionados. Además, se señala que, al tener sus ventajas o desventajas, un solo lenguaje no necesariamente cumple a cabalidad lo que se necesita.
- Toma de decisión: el contexto de software, el objetivo que se desee con el diagrama y además la experiencia previa juegan el papel preponderante para la elección.
- Característica relevante: complejidad, soporte de la comunidad, curva de aprendizaje y comprensión del lenguaje fueron mencionadas como las cualidades que poseen mayor peso a la hora de tomar la decisión.
- Característica no relevante: estandarizado y generación de código se mencionaron, esto debido a que lo que se necesita es que entre más sencillo el lenguaje es mejor. Además, muchas veces el código generado provoca un retrabajo mayor.

- Uso de los diagramas: comunicación con los diferentes niveles de negocio y técnico. Además, funciona para las diferentes fases del ciclo de desarrollo de software. Lo que es considerado más importante del diagrama es comunicar.

Administradores de Producto con mínimo 5 años

A continuación, se detallan las observaciones más relevantes de parte de los administradores de producto:

- Conocimiento de lenguajes visuales de modelado: BPMN y UML son los conocidos por este rol. Sin embargo, se menciona que han sido parte de la audiencia, y no han participado activamente en la creación propiamente de los diseños.
- Toma de decisión: a pesar de no estar activamente en la elaboración de estos, se menciona que los equipos han realizado bocetos sin un lenguaje en específico usando una notación conocida por todos los miembros de este.
- Característica relevante: holístico esto significa que se puedan representar no sólo ciertos niveles, sino también una representación para los diferentes tipos de audiencia que se puede llegar a tener. Además, es mencionada la flexibilidad en el sentido que se pueda hasta personalizar ciertos símbolos a usar, esto debido a que en el mundo actual el uso de proveedores ha crecido las integraciones (*CNCF Landscape*) es la manera de que se pueda crear un diagrama que contenga la información adecuada.
- Característica no relevante: Estandarizado, generación de código y generalidad son mencionadas como las cualidades que no pesan a la hora de elegir. Para profundizar en generalidad, hay lenguajes que no causan un impacto adecuado en la comunicación lo que provoca que se necesiten otros documentos que ayuden a la compresión de este.
- Uso de los diagramas: estos son utilizados para diferentes tipos de comunicaciones, entre las cuales se mencionaron: toma de decisiones, entendimiento del producto, priorización de trabajo, fechas de entrega entre otros.

En la tabla 11. *Resumen de aplicación de cuestionario a expertos*, se encuentran, de una manera puntual, las partes más relevantes que se obtuvieron de dichas entrevistas. Esto nos permite ver los patrones entre las diferentes conversaciones y obtener conclusiones sobre esta primera parte del diagnóstico.

Población/Característica	Conocimiento de Lenguajes	Toma de Decisión	Característica relevante	Característica no relevante	Uso de los diagramas
Desarrolladores	UML	Contexto del Software	Aprendizaje Flexibilidad	Estandarizado	Entendimiento de Flujos y Dependencias
Arquitectos	UML C4	Contexto del Software Objetivo del Diagrama Experiencia Previa	Complejidad Soporte de la comunidad Curva de Aprendizaje Compresión del Lenguaje	Estandarizado Generación del código	Comunicaciones técnicas y de negocio
Administradores de Producto	UML BPMN	Uso de bocetos en lugar de un lenguaje en especifico	Holístico Flexibilidad	Estandarizado Generación del código Generalidad	Toma de decisiones Entendimiento del producto Priorización Fechas de Entrega

Tabla 11. Resumen de aplicación de cuestionario a expertos
Fuente: Confeccionada por el autor

Estudio del Contexto de GodiTours

Como fue mencionado previamente, la idea del software por construir es brindar una experiencia inmersiva a los turistas que usan los servicios ofrecidos por GodiTours.

Aunque el enfoque de la organización ha sido totalmente en turismo, gracias a conocimientos obtenidos por GodiTours en relación con desarrollo de software y otros campos de la industria, ellos desean aventurarse a realizar el desarrollo de la aplicación de software que apoye su visión. Debido a su nivel de conocimientos y experiencia, ellos necesitan soporte en cuanto a la toma de algunas decisiones; dentro de las más relevantes está el lenguaje de modelado. En la Figura 9. *Generalidades Aplicación GodiTours*, se presenta información suficiente para comprender el contexto del software por desarrollar.

Figura 9. Generalidades Aplicación GodiTours
Fuente: Confeccionada por el autor

En la Figura 9 se omiten los requerimientos detallados, por razones de confidencialidad.

Además de conocer el contexto del software que se realizará, se estuvo conversando con el encargado de GodiTours en una entrevista similar a la realizada a expertos y analizada arriba. A continuación, se detallarán los resultados obtenidos de la entrevista. El detalle de la entrevista aparece en el Apéndice 4, *Información cuestionario a GodiTours*.

- Conocimiento de lenguajes visuales de modelado: solamente se conoce UML teóricamente, debido al conocimiento adquirido en la educación recibida.
- Característica relevante: el principal objetivo es facilitar la comunicación en distintos niveles, ya sea en conversaciones técnicas o de negocios. La capacidad de crear diagramas con diversas perspectivas es fundamental. Además, que el aprendizaje sea sencillo, pues no se puede invertir mucho tiempo para tal tarea.
- Característica no relevante: la generación de código es innecesaria debido a la naturaleza del software; se piensa que puede provocar demasiado retrabajo a la hora de la implementación.

- Uso de los diagramas: el fin es poder usar los diagramas no solamente en el área técnica, así como habilitar la toma de decisiones en el ámbito de negocio con base en mejores comunicaciones.

La información obtenida mediante la entrevista a GodiTours se resume en la Tabla 12. *Resumen de aplicación de cuestionario a GodiTours*.

Población/Característica	Conocimiento de Lenguajes	Característica relevante	Característica no relevante	Uso de los diagramas
GodiTours	UML	Holístico Curva de Aprendizaje	Generación de código	Conversaciones técnicas y de negocio Toma de decisiones

Tabla 12. Resumen de aplicación de cuestionario a GodiTours
Fuente: Confeccionada por el autor

Conclusiones de las entrevistas

Las entrevistas realizadas con los expertos y la organización nos permiten observar patrones o similitudes semejantes del tema bajo estudio. A partir de la información obtenida planteamos las siguientes conclusiones:

- **Perspectivas:** Los lenguajes visuales de modelado deben permitir perspectivas diferentes de acuerdo con el nivel de conversación que se quiere llevar a cabo, esto es, que el lenguaje permita realizar diagramas generales o específicos de acuerdo con la necesidad.
- **Curva de Aprendizaje:** Los lenguajes visuales de modelado deben tener una curva de aprendizaje baja, para permitir su uso y que la comprensión de estos sea sencilla. Básicamente que sin mucho tiempo se pueda intuir el significado de lo que se encuentra en un diagrama o modelo (dominio, reglas de negocio, diseño, etc.).
- **Estandarización no es necesaria:** A pesar de que la industria se esfuerza mucho en estandarizar los lenguajes visuales de modelado, al construir modelos van a tener carencias, lo que provoca el desuso de estos, por lo que se recurre a material de apoyo o bocetos rápidos que cumplen con el objetivo de acuerdo con el contexto de software.

- **Toma de decisión:** La decisión de un lenguaje visual de modelado se ve influida por el contexto del software, la experiencia previa de los miembros del equipo y el objetivo que se busca con los diagramas.
- **Generación de código es retrabajo:** La generación de código es algo irrelevante a la hora de tomar decisiones para la elección de un lenguaje de este tipo.

Análisis Externo

Moody (2009), en su estudio sobre la física de la notación en la Ingeniería del Software, menciona que una notación visual solo puede mejorarse cuando se comprende el cómo y por qué se comunica haciendo uso de este tipo de lenguaje. Además, se dan dos procesos complementarios: codificar y descodificar. La compresión se ve afectada por el ruido provocado, por ejemplo: los colores o símbolos que son usados.

En ese estudio, Moody (2009) propone nueve principios para mejorar la eficacia cognitiva de la notación visual en esta industria. Esta propuesta es usada para la creación de nuevos lenguajes, algunos de estos principios pueden ser usados para la elección de un lenguaje, pues permiten determinar las características que deberían poseer los lenguajes visuales de modelado. En la Tabla 13, *Principios para la eficacia cognitiva de la notación visual*, se hace una descripción de estos principios.

Principio	Descripción
Calidad Semiótica	Relación 1-1 entre los símbolos gráficos y constructores semánticos. Cada símbolo tiene un significado semántico.
Discriminación Perceptual	Los símbolos deben de ser distinguibles entre sí
Transparencia Semántica	Usar representaciones visuales cuya apariencia sugiere el significado.
Complejidad Administrativa	El modelo debe tener mecanismos que permitan ajustar la complejidad de acuerdo con la demanda. Modularidad, jerarquía, entre otros.
Integración Cognitiva	Si el proceso consiste en varios modelos, debería de haber maneras de incluir la información de otros modelos.
Expresividad Visual	Para incrementar la expresividad se debe usar el rango completo de variables visuales. Por ejemplo: color, textura, entre otros.
Codificación Dual	Modelo debe ser complementado con texto para dirigirse a ambos canales cognitivos.

Economía Gráfica	El número de símbolos debe estar limitado a un número cognitivo manejable.
Ajuste Cognitivo	Dependiendo de las circunstancias, el modelo debería ofrecer diferentes representaciones.

Tabla 13. Principios para la eficacia cognitiva de la notación visual
Fuente: Confeccionada por el autor basado en Moody (2009)

Koschmider, Drescher, & Lehner (2018) indican que, a pesar de que estos principios han sido usados en lenguajes actuales, se necesita una mayor investigación en algunos de ellos, entre los cuales se pueden mencionar:

- Transparencia Semántica: representación adecuada de las diferentes relaciones.
- Integración Cognitiva: mecanismos que ayuden al lector a armar la información de diferentes diagramas, simplificar la navegación entre los diagramas.
- Economía Gráfica: toma de decisiones sobre qué mostrar, o no, gráficamente.

Con la información recabada anteriormente, se puede observar que existen variables que pueden ser utilizadas para elegir un lenguaje visual de modelado. Sin embargo, la organización decide - según su contexto - qué peso tiene cada variable propiamente y si tiene sentido agregarla a la ecuación para la elección de un lenguaje versus otro(s).

Propuesta de Solución

La propuesta de la solución se basa en la información encontrada en el análisis de la situación, a fin de acoplar la herramienta a las necesidades de GodiTours.

A continuación, se describe la herramienta creada, tomando en cuenta los puntos relevantes de la indagación realizada tanto a nivel interno como externo. En las siguientes secciones se narra la conceptualización de la herramienta, así como las razones de las decisiones tomadas.

Definición de la Herramienta

El propósito de la herramienta es ayudar a la elección de un lenguaje visual de modelado con el fin de usarlo cuando se diseñe la arquitectura de las aplicaciones de software de GodiTours. Para cumplir con sus objetivos, la herramienta deberá poseer las características siguientes.

- **Flexible**: la herramienta debe ser flexible en caso de ser usada en diferentes tipos de proyectos, sin que se pierda su objetivo principal.
- **Documentada**: la herramienta debe contener la información de cómo debe ser usada. Las variables por ser usadas deben ser conceptualizadas de una manera adecuada.
- **Usabilidad**: la herramienta debe ser sencilla de usar. Su aprendizaje y su uso deben ser sin complicaciones innecesarias. El entendimiento de la herramienta debe ser eficaz, a fin de enfocar el tiempo en las fases siguientes, como el diseño del software.
- **Transparente**: la herramienta debe ser transparente en el sentido que las personas usuarias puedan comprender el trasfondo de esta, y la importancia que lleva la decisión.
- **Adaptable**: la herramienta debe poder ser adaptada al contexto donde se desarrolla el software, por lo que puede ser ajustada de acuerdo con las necesidades que se tenga a su momento de uso.

Toma de Decisión

La importancia de la herramienta se basa en las variables que serán usadas para generar una elección adecuada, de acuerdo con el contexto y sistema de la organización, como se puede observar en la Figura 10. *Resumen de características de la investigación*, se encuentra la lista de las características que fueron relevantes y cuáles no, tanto para la organización como para los entrevistados (junto al análisis externo).

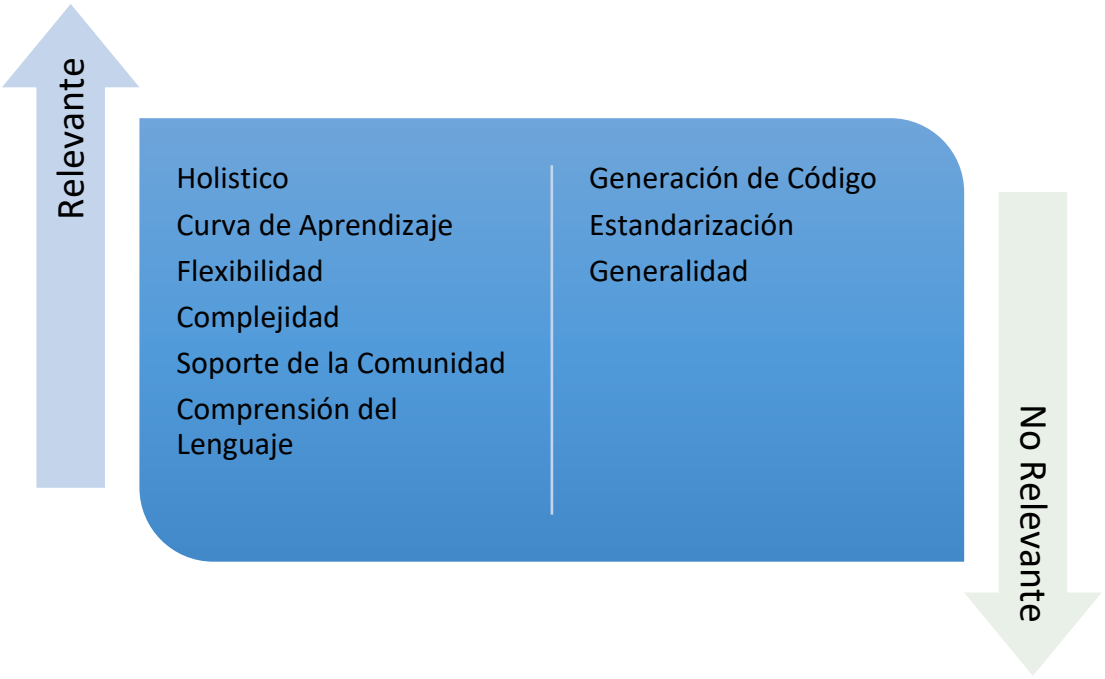


Figura 10. Resumen de características de la investigación
Fuente: Confeccionada por el autor

Las características que son tomadas en cuenta para la herramienta son aquellas que se encuentran mencionadas como relevantes (ver el lado izquierdo de la figura). Las características contarán con palabras de un entendimiento sencillo, para evitar complicaciones al usar la herramienta.

Además, se evita hacer un uso extensivo de características pues esto haría más complicada la elección.

Características de la Herramienta

La herramienta va a contar con las siguientes secciones:

- Información General: contiene información general del proyecto, como nombre, objetivo del proyecto, personas involucradas en las discusiones.
- Matriz de Variable: contiene los lenguajes y las variables a ser evaluadas con el fin de obtener el lenguaje que mejor se adecue al contexto.

- Resultado de elección: la elección debe quedar documentada para posibilitar futuras conversaciones.
- Comentarios: sección donde se puede colocar información relevante sobre la discusión para el uso de la herramienta que sirva para futuros proyectos de software.

En la tabla 14. *Definiciones de características a usar en la herramienta de elección*, se definen las variables con las cuales se evaluarán los diferentes lenguajes de modelado. Además, se detalla cuál será la escala de evaluación de cada variable, con el fin de que el conocimiento previo de algún lenguaje no tenga peso en la elección.

Característica	Definición	Escala de Evaluación
Holístico	Esta característica persigue comprender si el lenguaje visual permite ver las diferentes aristas y vistas que tiene la arquitectura de software, así como una integración cognitiva adecuada. Se busca permitir una comunicación eficiente con los diversos involucrados del proyecto.	Alta: Posee más de 3 diagramas Media: Posee 2 diagramas Baja: Posee un diagrama
Curva de Aprendizaje	Se desea que la curva de aprendizaje sea reducida debido al conocimiento que tiene la organización. No se puede usar mucho tiempo aprendiendo un lenguaje con este fin.	Alta: Mayor a un mes Media: Entre dos semanas y un mes Baja: Menor a dos semanas

Flexibilidad	En cuanto a la transparencia semántica y economía gráfica, el lenguaje debe permitir hacer uso de símbolos gráficos propios o de terceros, por ejemplo: logos de *cloud native landscape*, entre otros.	Alta: Uso de diferentes símbolos propios o de terceros Media: Uso de símbolos de terceros Baja: Uso solo de la notación dictada
Documentación	La información disponible y el soporte de la comunidad es suficiente para entender el lenguaje, además de clarificar información en caso de ser necesario	Alta: Información disponible en más de 10 sitios Media: Información disponible entre 5-10 sitios Baja: Información disponible en menos de 5 sitios

Tabla 14. Definiciones de características a usar en la herramienta de elección
Fuente: Confeccionada por el autor

Elaboración de la Herramienta

La tabla 15. *Herramienta para la elección*, ilustra la forma que tiene la herramienta real, creada en una hoja de cálculo, para ofrecer a la organización algo de uso sencillo y fácil de compartir entre los miembros.

Información General	
Nombre: **Objetivo del Proyecto** **Metodología:** **Personas involucradas**	

Lenguaje/Variable	Holístico	Curva de Aprendizaje	Flexibilidad	Documentación

Resultado

Comentarios

Tabla 15. Herramienta para la elección
Fuente: Confeccionada por el autor

Los lenguajes son seleccionados de una lista, al igual que la evaluación con las variables. Conforme se va completando cada variable de acuerdo con la selección, se aplica un código de colores de acuerdo con la escala de evaluación que permite expresividad visual, además de seguir uno de los principios propuestos por Moody (2009). Con esto, al equipo que use la herramienta, se le hará sencillo ir conociendo la selección y la evaluación que se está realizando. Para un mejor entendimiento de esto, en la siguiente tabla simula un uso de la herramienta.

Información General	
Nombre:	**Simulación**
Objetivo del Proyecto	Simulación del uso de la herramienta
Metodología:	Híbrida
Personas involucradas	Juan, Miguel, Andres, Didier

Lenguaje/Variable	Holístico	Curva de Aprendizaje	Flexibilidad	Documentación
UML	Media	Media	Alta	Media
Togaf	Alta	Baja	Baja	Alta
BPMN	Media	Alta	Baja	Alta
SysML	Media	Baja	Media	Baja
ArchiMate	Baja	Baja	Media	Alta
C4	Alta	Media	Alta	Media

Resultado	**Togaf**
Comentarios	Simulación del uso de la herramienta

Tabla 16. Simulación de la herramienta
Fuente: Confeccionada por el autor

Aplicación de la Herramienta

Para hacer uso de la herramienta, se elaboró un proceso con el fin de que la introducción y empleo de la herramienta sea adecuado, además de reducir o prevenir riesgos que provoquen resultados erróneos.

En la Figura 11. *Proceso de Implementación de Herramienta*, se observan las fases del proceso, así como el objetivo, actividades y entregables de estas. A continuación, se explica la razón de cada fase y su funcionalidad en cada etapa o paso.

- **Pre-implementación:** la idea es que el equipo pueda tener un conocimiento general de los lenguajes visuales, además de la herramienta y las variables, para ellos se usará información de esta investigación.

- *Lecturas*: se compartirá información general de los lenguajes, además de la herramienta para que el equipo la lea y pueda tener ese conocimiento antes de hacer uso de la herramienta.
- *Reunión corta de seguimiento*: de ser necesario se hará una reunión con el equipo para aclarar dudas y/o comentarios de acuerdo con la información compartida, con ello se evitará perder el enfoque al usar la herramienta.

- **Implementación:** esta fase consiste en realizar el uso de la herramienta.
  - *Reunión de implementación:* junto con los miembros del equipo se hará una reunión para hacer uso de la herramienta y así poder tener la elección de lenguaje visual de modelado por usar.
- **Post-implementación:** última fase del proceso, con el fin de obtener información sobre el uso de la herramienta para poder hacer un análisis sobre la implementación de esta.
  - *Entrevistas:* con los miembros del equipo, se tendrán conversaciones abiertas para conocer su parecer respecto de la herramienta y los puntos por mejorar de esta.

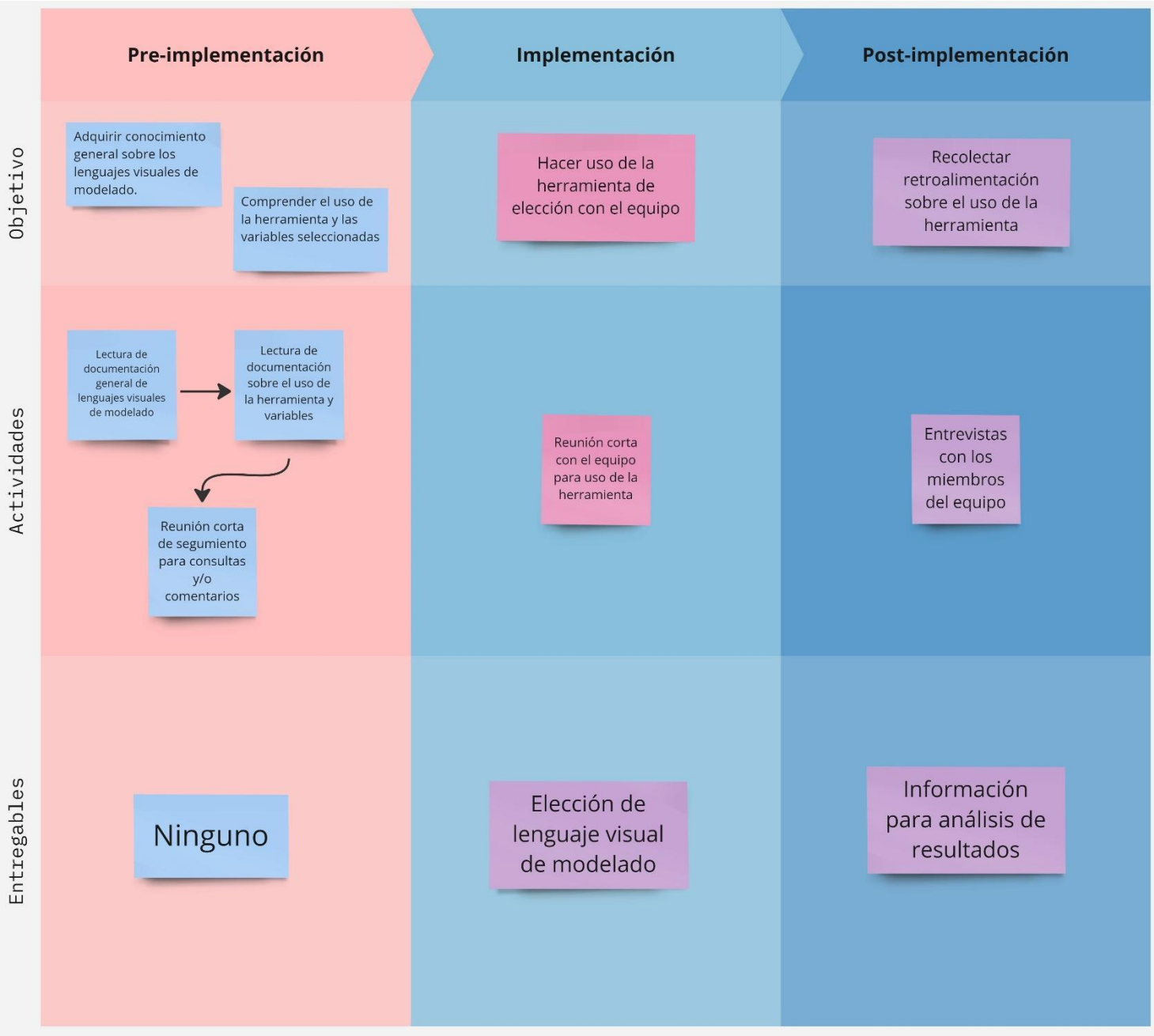


Figura 11. Proceso de Implementación de Herramienta
Fuente: Confeccionada por el autor

Resultados Obtenidos de la Herramienta

De acuerdo con el proceso explicado en la sección anterior se procede a detallar los resultados obtenidos en su implementación:

- **Pre-implementación:**

- *Lecturas*: las lecturas fueron enviadas por medio de un correo electrónico; la información compartida fue mayoritariamente la usada en esta investigación.
- *Reunión corta de seguimiento*: en la reunión se conversaron temas propios de la herramienta, el uso de esta y consultas generales que ayudaron a reforzar las definiciones y escalas de las variables seleccionadas.

- **Implementación:**
  - *Reunión de implementación:* además de hacer uso de la herramienta, en esta reunión se obtuvo información para mejorarla ante posibles usos en futuros proyectos. En la tabla 17. *Herramienta completada junto a GodiTours*, se logra observar que el lenguaje elegido para la experiencia interactiva es **C4** pues es la permite cumplir mejor con los objetivos planteados previamente.

Información General	
Nombre:	**Experiencia Interactiva GodiTours**
Objetivo del Proyecto	Crear una experiencia inmersiva en nuestros servicios de turismo
Metodología:	Ágil
Personas involucradas	Juan, Diego, Leo, Steven

Lenguaje/Variable	Holístico	Curva de Aprendizaje	Flexibilidad	Documentación
UML	Alta	Alta	Baja	Alta
ArchiMate	Alta	Alta	Baja	Media
BPMN	Alta	Alta	Baja	Media
C4	Alta	Media	Alta	Alta

Resultado	C4

Comentarios	La discusión fue basada no solamente en la información que habíamos encontrado. Se investigó las nuevas tendencias de la industria. Fue necesario modificar la escala debido a cómo se iba dando la discusión y lo que se iba entendiendo de cada punto de vista expresado en la reunión.

Tabla 17. Herramienta completada junto a GodiTours
Fuente: Confeccionada por el autor

- **Post-implementación:**
    - *Entrevistas:* a continuación, se destacan los puntos importantes que se obtuvieron en las entrevistas realizadas a los presentes en la reunión. La información completa de estas aparece en el Apéndice 5, Información *entrevista post-implementación*.
        - Material brindado: la información compartida antes de la reunión ayudó a que los miembros estuvieran en la misma página para el uso de la herramienta y la discusión se enfocó en llegar a un acuerdo en la escala.
        - Discusión provechosa: la discusión que se tuvo en la reunión fue adecuada pues se dieron a conocer puntos de vista diferentes antes de llegar a acuerdos y se permitió una mayor cohesión del equipo de trabajo.
        - Enfoque puede variar: la forma de usar la herramienta se puede modificar, en el sentido de que cada individuo la llene separadamente, y se discuta solamente sobre las diferencias encontradas, para facilitar la participación de personas que no suelen conversar en grupos grandes.
        - Facilidad de uso: la herramienta es sencilla de usar, no se encontraron complicaciones en ellas, la tonalidad de colores permite un entendimiento sobre lo que es o no adecuado. Además, gracias a las lecturas, las definiciones facilitar la selección de escalas adecuadas al contexto.

Análisis de los Resultados Obtenidos

Los resultados son analizados para determinar el cumplimiento de los requerimientos planteados en la definición de la herramienta revisada. A continuación, se detallará cada uno:

- **Flexible**: debido a la discusión que se tuvo, fue necesario modificar la escala a usar ya que reflejaba cierto sesgo debido a conocimiento previo y la información que se compartió. La escala se modificó en ciertas variables en cuanto a valores, para que tuviera sentido según el contexto de los presentes en la reunión. Para futuros usos, además de las variables, se deberá considerar las escalas por usar con el equipo de trabajo. Se pudo observar la flexibilidad de la herramienta, que pudo ser modificada para adecuarla a las necesidades.
- **Documentada**: la documentación que se compartió incluyó a la herramienta y a los lenguajes por evaluar. Esto ayudó a que el equipo de trabajo pudiera enfocarse en

una discusión de valor y no perdiera el tiempo en otros aspectos innecesarios para el uso de la herramienta. Los participantes comentaron que la herramienta es fácil de usar y es intuitiva.

- **Usabilidad**: los comentarios respecto de la herramienta en cuanto a la tonalidad de los colores, los campos seleccionados, variables y uso en general permite que la carga cognitiva sea reducida cuando se está analizando la información, lo que enfoca la discusión en la evaluación de los lenguajes.
- **Transparente**: aunque este requerimiento suele ser un poco abstracto, en las entrevistas se logró obtener la información adecuada para evaluarlo. Después del uso de la herramienta el equipo de trabajo comentó que ahora comprenden por qué llegar a un acuerdo sobre cuál lenguaje visual de modelado usar es algo de importancia, y cómo les ayudará a cumplir con los objetivos planteados para el software por desarrollar.
- **Adaptable**: la adaptabilidad de la herramienta es una de las características más mencionadas pues fue necesario adecuarla mientras se tenía la discusión; agregar variables u otros lenguajes se puede hacer de manera sencilla.

En resumen, los resultados obtenidos son favorables debido a que la herramienta cumplió su objetivo principal de facilitar la elección de un lenguaje visual de modelado de acuerdo con el contexto presentado por GodiTours. Además, permitió que el equipo de trabajo tuviera un mejor entendimiento en las razones por las cuales los diseños son de gran ayuda para el desarrollo de un sistema de software - pues habilitan una mejor comunicación en los diferentes niveles.

Documentación de la Arquitectura

Proceso de Creación de la Arquitectura

Luego de elegir a C4 como lenguaje visual de modelado por usar para el diseño de la arquitectura, gracias a la herramienta, se empieza con las primeras fases del ciclo de vida de desarrollo de software de una manera ágil. La organización necesita obtener un prototipo que permita ese punto de entrada para establecer la aplicación y conseguir no solo conversaciones, sino posibles inversiones para el desarrollo completo de esta.

En la Figura 12. *Proceso de Creación de la Arquitectura se* observa cómo se llevó a cabo este proceso para atender la necesidad. El proceso está basado en el enfoque de arquitectura continúa descrito por Erder y Pureur (2015), cuyo enfoque está basado en los siguientes principios:

- **Arquitecturas para productos:** cambiar el enfoque de trabajar en soluciones específicas para proyectos, a trabajar sobre productos, ya que permiten una forma de trabajar más eficiente y priorizar las necesidades de los clientes.
- **Enfocarse en atributos de calidad:** los atributos de calidad dirigen el tipo de arquitectura requerida.
- **Retrasar decisiones de diseño:** no se debe gastar recursos en diseñar e implementar funcionalidades que tal vez no lleguen a ser usadas. Por eso se debe retrasar las decisiones lo más que se pueda para que sean basadas en hechos y no en supuestos.
- **Arquitecturas preparadas para el cambio:** usar el beneficio de trabajar a menor escala y en componentes que no estén acoplados, esto brinda muchas ventajas a la hora del desarrollo de software.
- **Arquitecturas para construir, probar, desplegar y operar el software:** las buenas prácticas actuales de la industria para permitir una entrega continua de software deben ser tomadas en cuenta a la hora diseñar su arquitectura.
- **Modelar la organización de los equipos:** la organización de los equipos dirige la arquitectura y el diseño de los sistemas (ley de Conway), por lo que una vez diseñado el software se requiere que la organización sea modelada de acuerdo con las necesidades de este.

Para poder entender el contexto de dicho proceso de creación, primero se detalla cómo está organizado el equipo de trabajo a nivel de roles. Debido a la naturaleza del proyecto y organización algunos roles pueden ser ejecutados por una misma persona. La organización está basada de acuerdo con lo detallado por Schwaber y Sutherland (2020), los creadores de Scrum. En la siguiente tabla se encuentra una breve descripción de cada rol, desde dos puntos de vista: el primero es tomando en cuenta en todo el ciclo de desarrollo de software y el segundo es enfocado en el papel que cada rol tiene en las discusiones para la creación de la arquitectura.

Rol	Desarrollo de Software	Discusión de Arquitectura
Administrador de Producto	Encargado de tener la visión del producto y estrategia de este. Comprende cuál es el mejor camino por seguir en cuanto a características esenciales y prioridades a seleccionar. Además, logra traducir de una manera adecuada las necesidades del cliente ante el grupo de trabajo.	Este rol permite verificar y validar que los atributos relevantes para el usuario se encuentren presentes en la arquitectura. Además, ayuda al equipo a enfocar la solución a las prioridades dadas.
Líder técnico	Encargado de velar por el área técnica del equipo, recomienda tecnologías por usar, patrones de diseño, buenas prácticas y decisiones que permiten cumplir con las solicitudes del cliente o el usuario final del producto.	El rol que da la decisión final si no se llega a un acuerdo en el equipo, pone la experiencia al servicio del equipo. Además, ayuda a traducir en ambas vías: del negocio y técnica.
Desarrollador	Persona a cargo de llevar a cabo la construcción del software, también participa en la toma de decisiones de acuerdo con su conocimiento y experiencia.	Aplica las prácticas actuales de la industria para implementar y solucionar las necesidades del cliente. Su punto de vista es importante porque contempla todas las fases del desarrollo de software al hacer realidad la arquitectura.

Tabla 18.. Equipo de trabajo y puntos de vista
Fuente: Confeccionada por el autor

Las decisiones tomadas fueron acuerdos por todo el equipo de trabajo. Ninguna decisión fue tomada por un solo rol, para exponer y entender los diferentes puntos de vista y poder tener una mejor toma de decisión, con base en los principios de arquitectura continua. Además, las características se describen de manera general, por estar más allá del alcance de esta investigación. El detalle es suficiente para comprender el contexto del proyecto en desarrollo.

A continuación, se detalla las fases del proceso utilizado para el diseño de la arquitectura usando el lenguaje seleccionado:

- **Análisis de Requerimientos:** los requerimientos de la aplicación fueron analizados por el equipo de trabajo, y se aclararon posibles dudas o zonas grises. Se logró una mejor comprensión del objetivo de la aplicación por todos los miembros.
- **Selección de Requerimientos:** el equipo de trabajo priorizó los requerimientos con base en la relevancia de los flujos para el usuario final. Asimismo, este mejor entendimiento de la aplicación facilitará futuras conversaciones con posibles inversionistas.
- **Análisis de Alcance:** con los requerimientos priorizados, se analiza el alcance que tendría el producto con estos, manteniendo sus objetivos y que sea un producto funcional.
- **Diseño:** se realiza el diseño que cumpla con los requerimientos que fueron seleccionados, analizar el esfuerzo que conlleva mucho esfuerzo por parte del equipo de trabajo y se tiene que iterar en la selección de requerimientos y análisis de alcance. Luego de realizado el diseño se usa una lista de verificación publicada en el sitio oficial de C4, con el fin de validar si los diagramas siguen las recomendaciones y buenas prácticas asociadas.

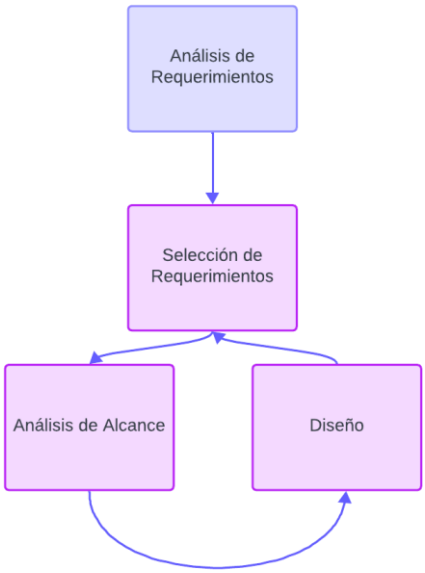


Figura 12. Proceso de Creación de la Arquitectura
Fuente: Confeccionada por el autor

El motivo de que el proceso de creación de la arquitectura sea iterable es obtener diseños que se adecuen a la necesidad del contexto sobre el cual se está realizando el proyecto.

Notación y Artefactos del Diseño de la Arquitectura

Notación

Una de las características del lenguaje visual de modelado C4 es su independencia respecto de la notación por usar. A diferencia de otros lenguajes visuales de modelado, que describen y detallan todos los aspectos sobre cómo se define el lenguaje, C4 parte de una notación simple que permite el entendimiento entre las partes. En C4 bastan los bocetos, usando tarjetas u otros instrumentos que permitan la creación de los diferentes niveles.

En el sitio oficial de C4 se prescriben ciertas reglas generales que debería de tener la notación por usar:

- La notación por usar se tiene que autodescribir, para que la carga cognitiva no sea pesada para las personas que usan los diagramas.

- Todos los niveles (diagramas) deben de tener leyendas para hacer la notación explícita; así, las personas se enfocan en el contenido y no en cómo deben entender el diagrama.

Además de esto, ofrece varias recomendaciones o buenas prácticas que deben considerarse con el fin de proceder a realizar diagramas usando este lenguaje de una manera adecuada. Estas se describen en la Tabla 19. *Recomendaciones C4*

Diagramas	**Elementos**	**Relaciones**
Cada diagrama debe de tener un título descriptivo del tipo y alcance	El tipo de cada elemento debe ser explícitamente explicado	Cada línea representa una relación unidireccional
Cada diagrama debe tener una leyenda que explique la notación por ser usada (formas, colores, líneas, entre otros)	Cada elemento debe poseer una descripción corta que provee una vista general de responsabilidades	Cada línea debe tener una etiqueta específica que sea consistente con la dirección e intención de la relación
Si se usan abreviaciones o acrónimos, estos deben ser comprendidos por la audiencia o explicadas en el diagrama	Cada contenedor y componente debe tener una tecnología explícitamente especificada	Las relaciones entre contenedores deben tener una tecnología/protocolo en la etiqueta,

Tabla 19. Recomendaciones C4
Fuente: Confeccionada por el autor con datos de The C4 model for visualising software architecture

Detallada toda esta información, a continuación, se explica cuál es la notación seleccionada para la creación de los diferentes diagramas de la arquitectura, con el fin de que los involucrados logren comprender de una mejor manera estos documentos. Aunque C4 no posee una notación estandarizada, cuenta con ejemplos en su sitio web: la notación fue creada para este proyecto será usada en forma comparativa con la del sitio Web. Se podrán ver las diferencias entre ambas notaciones. Se proveen ejemplos de ambas para observar cómo se usa cada una.

En la Tabla 20. *Notación comparativa C4 y GodiTours*, se puede ver que la única diferencia importante es en la notación para el contenedor. Este cambio se da debido como resultado de la primera iteración del proceso de diseño de la arquitectura. El equipo de trabajo agrega dentro del contenedor los iconos o imágenes del servicio o tecnología por utilizar (cuando sea necesario) para facilitar el entendimiento del diagrama.

Artefactos

Además de lo relevante que es especificar la notación, hay varios artefactos mencionados por Woods (2023) basados en el enfoque de arquitectura continua y sus principios, que son usados para comprender el contexto y las justificaciones en las diferentes tomas de decisiones. Estos deben tener una audiencia y propósito para que sean útiles, asimismo deben tener las siguientes características de acuerdo con Woods (2023):

- **Mínimo**: debe contener solamente la información que sea relevante según la audiencia y el objetivo que posee el documento.
- **Usable**: debe poder ser utilizado por la audiencia para algún motivo, ya sea para toma de decisiones o meramente informativa.
- **Significante**: para la audiencia debe de llamar la atención.
- **Indexado**: debe de ser de fácil acceso y que pueda encontrarse de una manera sencilla.
- **Revisado**: debe pasar por una revisión para asegurar que cumpla con el propósito propuesto.

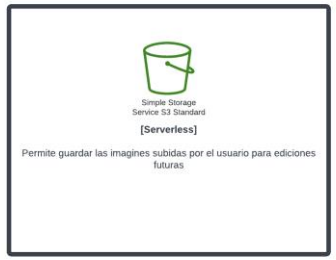
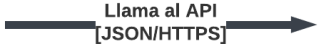
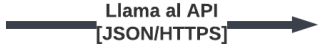
Notación	C4	Ejemplo C4	GodiTours	Ejemplo GodiTours
Usuario				
Contenedor				
Relación				

Tabla 20. Notación comparativa C4 y GodiTours
Fuente: Confeccionada por el autor

Dentro de estos artefactos se seleccionan dos que serán usados debido a su conceptualización y al contexto del proyecto. A continuación, se detallan y se especifica el estilo de plantilla que van a tener estos:

Principios

Uno de los artefactos que es muy usado y que con el uso de las metodologías ágiles se popularizó, es el *team agreement* o acuerdos de equipo. De acuerdo con Kitch (2023) este describe las expectativas, roles y responsabilidades del equipo y comúnmente es creado por todos los miembros del equipo. Además, Kitch menciona cuáles cosas pueden ser cubiertas por este:

- Cómo deben tomarse las decisiones
- Proceso para manejar el conflicto
- Expectativas de comunicación
- Métricas para medir el éxito de cada proyecto

Como se puede observar, básicamente son aquellos principios para que el trabajo en equipo se pueda realizar de una manera eficaz y eficiente. Además, da una guía de trabajo para cada miembro del equipo.

Dentro del enfoque de arquitectura continua, los principios suelen ser aquellos que permiten tener una guía al equipo para que las decisiones que se tomen estén alineadas y los miembros puedan saber y reconocer cómo tomarlas. Se puede tomar como un anexo al acuerdo del equipo, pero enfocado en las decisiones de arquitectura o un artefacto totalmente separado.

Para GodiTours, debido a que el acuerdo del equipo está fuera del alcance de este proyecto, se hizo como documento separado. Al igual que se hizo con la notación, se detalla la plantilla que es usada por el equipo de trabajo.

En la Figura 13. *Plantilla Principios del Equipo* se puede observar cómo es esta plantilla, la idea es seguir las recomendaciones mencionadas anteriormente, para favorecer el uso del artefacto.

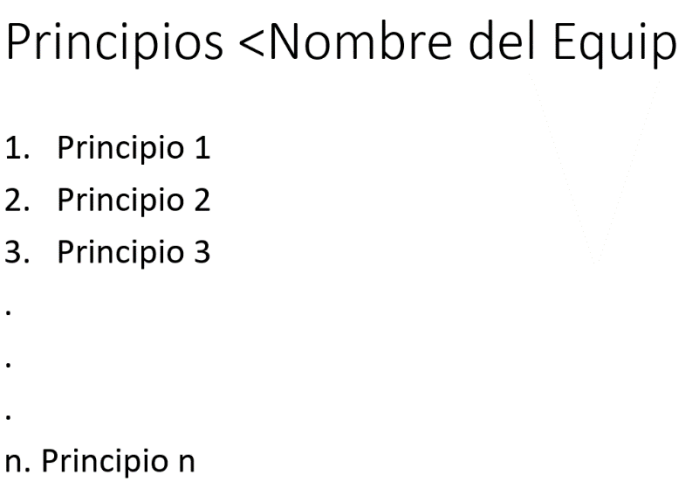


Figura 13. Plantilla principios del equipo
Fuente: Confeccionado por el autor

Además, al uso de la plantilla se da las siguientes recomendaciones al equipo de trabajo, con el fin de sacar el máximo provecho a esta:

- Al igual que un acuerdo de equipo, los principios deben acordarse con la mayoría del equipo de trabajo.
- La redacción de los principios debe ser clara, precisa y concisa, para evitar la ambigüedad.
- Se deben revisar los principios cada cierto tiempo con el fin de que estos se actualicen de acuerdo con las necesidades del equipo y de la organización.
- Los principios deben ir enfocados a cómo deben ser dirigidas las decisiones sobre los diseños de arquitectura, y a las comunicaciones sobre este tema - nada más.
- El archivo debe ser guardado en un lugar que sea accesible para los miembros del equipo, con el fin de garantizar el acceso.

Decisiones

Debido a la naturaleza del enfoque de arquitectura continua, donde las decisiones son tomadas constantemente por el equipo de trabajo, una práctica recomendada por Woods (2023) es

la documentación de estas, para que el equipo no solamente entienda las razones en las cuales se basaron las decisiones, sino tener un histórico de ellas. Esto permite un entendimiento del software y de su contexto. Dentro de la importancia de documentar las decisiones se tiene:

- Imparcialidad: el documentar provoca que la persona piense sobre la decisión que está tomando y tome en cuenta otros aspectos, no solamente basarse en lo que la persona conoce y entiende.
- Razonamiento: entender en qué se basó la decisión, en cuanto a contexto, alternativas y la propia solución.

Como es conocido por la industria de software, muchas veces las personas a cargo de tomar las decisiones sobre el diseño lo hacen con base en su experiencia e intuición. Según Tang y van Vliet (2009) esto causa que no haya un proceso adecuado para entender el contexto de la decisión y provoque no entender el razonamiento detrás de esto, sino que a futuro el mantenimiento y la operación del software se vuelvan complicados.

Van Heesch, Avgeriou, & Hilliard, (2012) mencionan cómo las decisiones y el proceso racional de estas debe ser documentado e integrado a la documentación que se genere en la fase de diseño de software, con el fin de proveer trazabilidad entre las decisiones que se tengan que tomar.

Por esas razones es relevante para el equipo documentar y tener accesibles las decisiones que se toman al diseñar, para permitir no solamente un proceso de desarrollo adecuado, sino que en fases posteriores se puedan consultar para - por ejemplo - determinar el impacto que podrían tener las modificaciones sobre decisiones tomadas previamente.

Woods (2023) menciona que los puntos mínimos a contener en este tipo de documentación son los siguientes:

- Título: título que permita entender el contexto de la decisión de una manera rápida y sencilla.
- Descripción de la decisión: permite entender cuál fue la decisión tomada, sin adentrarse en la justificación de esta.

- Razonamiento de la decisión: entender cómo se tomó la decisión y basado en qué, con el fin de comprender contexto, alternativas y solución.
- Fecha de la decisión: cuándo fue tomada la decisión.

Este tipo de plantilla contiene lo necesario para que los miembros del equipo actuales y futuros entiendan la justificación de las razones. Sin embargo, en la parte de razonamiento de la decisión, no se detalla qué partes la componen y cómo puede ser desplegada la información. Por ello, se plantea una plantilla un poco diferente que se puede observar en la Figura 14. *Plantilla de la decisión*, detallada a continuación:

- **Fecha de la decisión:** cuando fue tomada la decisión.
- **Título:** título que permita entender el contexto de la decisión de una manera rápida y sencilla.
- **Descripción de la decisión:** permite entender cuál fue la decisión tomada, sin adentrarse en su justificación.
- **Integrantes:** personas que estuvieron en la toma de la decisión.
- **Razonamiento de la decisión:** basado en Tang, van Vliet (2009) y Van Heesch, Avgeriou, & Hilliard, (2012), proponen los términos de preocupaciones y fuerzas, que se pueden definir ambos como toda aquella información, situación o entrada que tiene influencia sobre el diseño. Para evitar confusiones, se hace una matriz que posee la razón y las alternativas, con su escala respectiva para que se pueda evaluar adecuadamente.

<Fecha>

<Título>

Descripción de la decisión: <Información de la decisión tomada>

Integrantes: <Miembros del equipo que estuvieron en la toma de la decisión>

Razonamiento de la Decisión:

Preocupación / Fuerza	Alternativa 01	Alternativa 02	Alternativa 03
P / F 01	Soporta	Medio	Neutral
P / F 02	Opuesta	Soporta	Soporta
.	.	.	.
P / F n	Soporta	No Soporta	Neutral

Figura 14. Plantilla de la decisión
Fuente: Confeccionada por el autor

Diseños de la Arquitectura

En esta sección se muestran los diseños que fueron creados para la versión seleccionada por GodiTours. Adicionalmente, ofrece una descripción del contenido de cada uno. Las decisiones y justificaciones se detallan en la siguiente sección, con el fin de enfocarse mejor en cada aspecto. A continuación, se mencionan los puntos importantes de dicho diseño:

- Aplicación en la nube: la aplicación será creada en la nube debido a las diferentes ventajas que ofrece a nivel de infraestructura y servicios.
- Arquitectura de microservicios: para poder cumplir con las diferentes funcionalidades el estilo de la arquitectura es de microservicios, permitiendo agregar, eliminar o actualizar servicios independientemente de acuerdo con la necesidad del sistema.
- Tecnología Serverless: para permitir un desarrollo más efectivo y eficiente, se hará uso de servicios serverless con el fin de evitar también el usar recurso humano para mantener infraestructura.

Vista del Contexto

Esta vista permite entender el contexto donde se encuentra el sistema de software y si este se relaciona con otros. Al ser una aplicación totalmente nueva y debido a la naturaleza de esta, por el momento no hace uso de otros sistemas.

Por esto, el diagrama contiene solamente como contexto al usuario (turista) que hará uso del sistema. Esto se puede ver en la Figura 15. *Vista del Contexto del Sistema*.

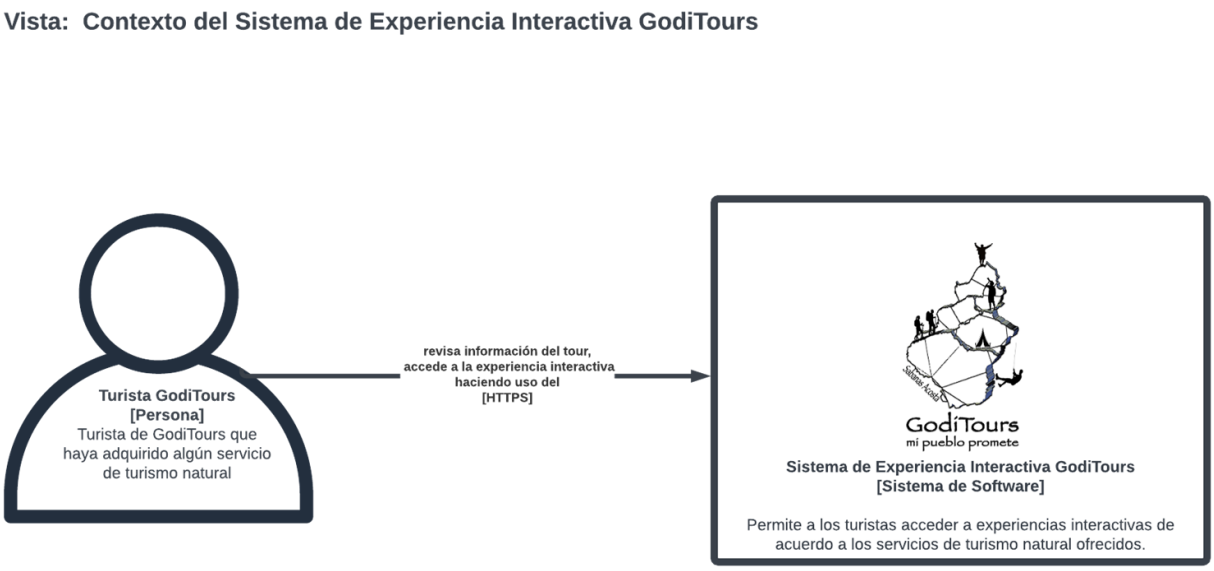


Figura 15. Vista del contexto del sistema
Fuente: Confeccionado por el autor

Vista de Contenedores

En esta vista podemos ver estará compuesta la arquitectura en términos de contenedores, los cuales funcionarán en conjunto para cumplir con los objetivos:

- Aplicación Móvil: este será el punto de entrada del usuario, brindará acceso y autorización para hacer uso de la aplicación y los diferentes servicios que se poseen.
- Aplicación API: esta aplicación se encargará de brindar toda la información necesaria a los usuarios además de crear toda la experiencia para el turista.

- Base de Datos: en este contenedor se tendrá todos los datos para que la funcionalidad de la aplicación se pueda llevar a cabo.

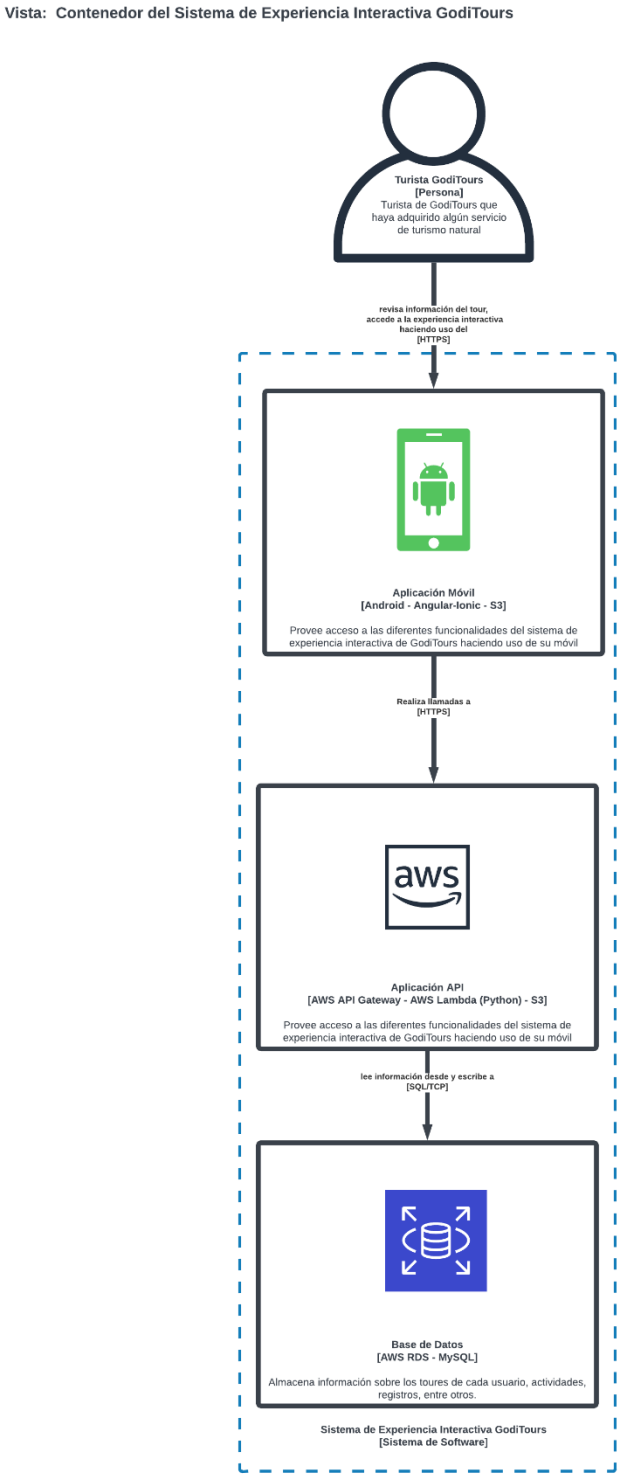


Figura 16. Vista Contenedor del Sistema
Fuente: Confeccionada por el autor

Vista de los Componentes del Sistema

Esta vista permite observar cómo los componentes se comunicarán entre sí y en qué estarán basados. En la Figura 17. *Vista de los Componentes del Sistema*, se observa con detalle cómo se encuentra separada cada aplicación:

- Aplicación Móvil:
    - Aplicación Web Estática: provee el acceso a la aplicación y que el usuario haga uso del sistema.
    - Administración de Sesiones de Usuarios: permite a los usuarios acceder al sistema, provee además la autorización a estos.
- Aplicación API:
    - Tours: acceso a los tours seleccionados por parte del usuario y la experiencia interactiva.
    - Experiencia: acceso a la experiencia interactiva, encargada del manejo del hardware, conexión con puntos de interés, entre otros.
    - Mapas: acceso a las rutas de los diferentes tours creados, además de la integración con mapas reales.
    - Notificaciones: provee notificaciones para los demás componentes.
    - Descuentos: acceso a los descuentos de los negocios incluidos en el sistema para que puedan ser usados por los turistas.
    - Información: acceso a información general de GodiTours.

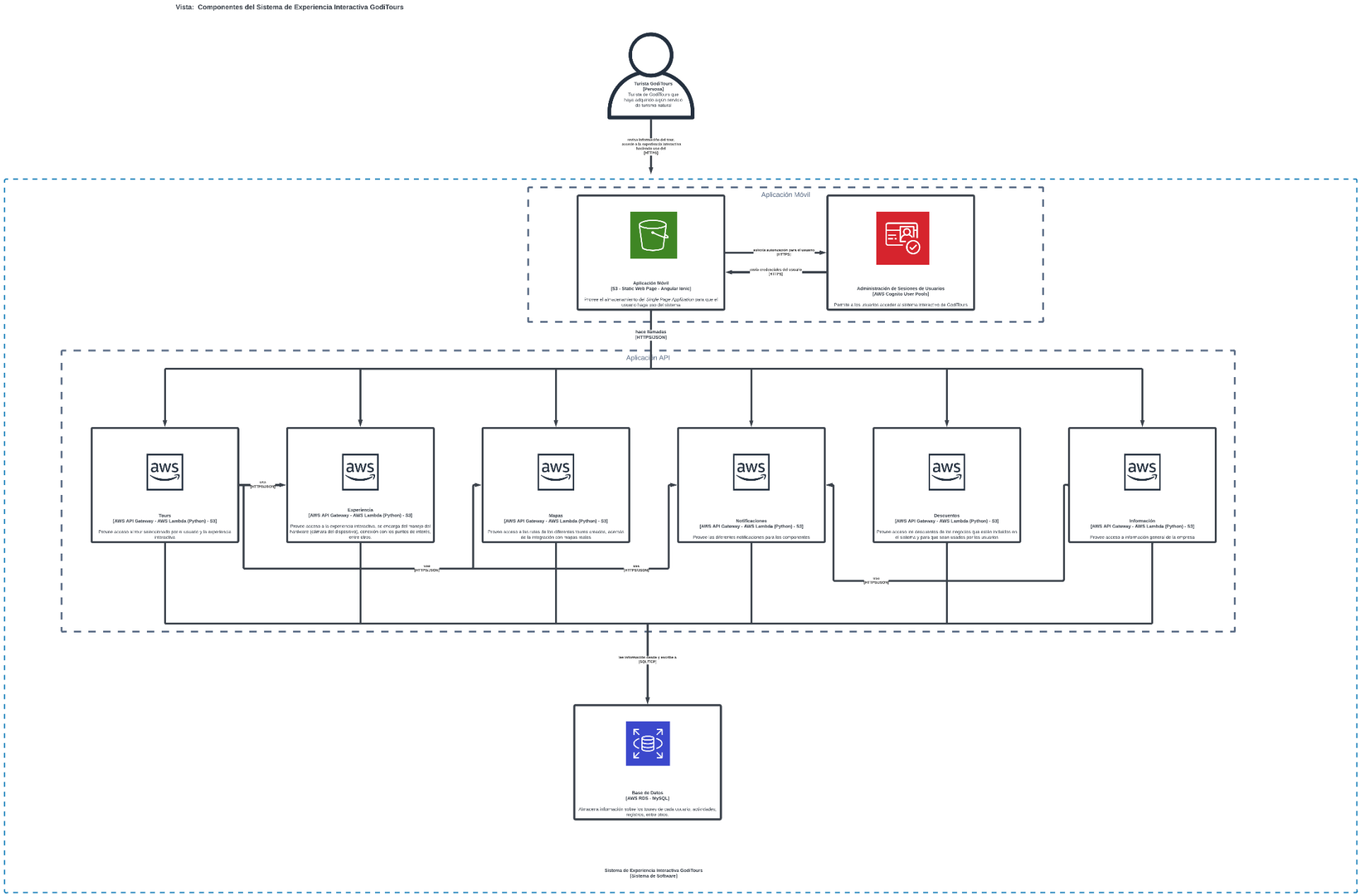


Figura 17. Vista de los componentes del sistema
Fuente: Confeccionado por el autor

Vista de Código del Sistema

En esta vista el diseño es un poco diferente a lo documentado en C4, esto debido a que en esta fase el código aún no es necesario, por lo tanto, el propósito de este es el de entender el flujo que se va a tener con los diferentes servicios de la nube, debido a que ayuda a comprender mejor el sistema como tal.

Vista: Código del Sistema de Experiencia Interactiva GodiTours

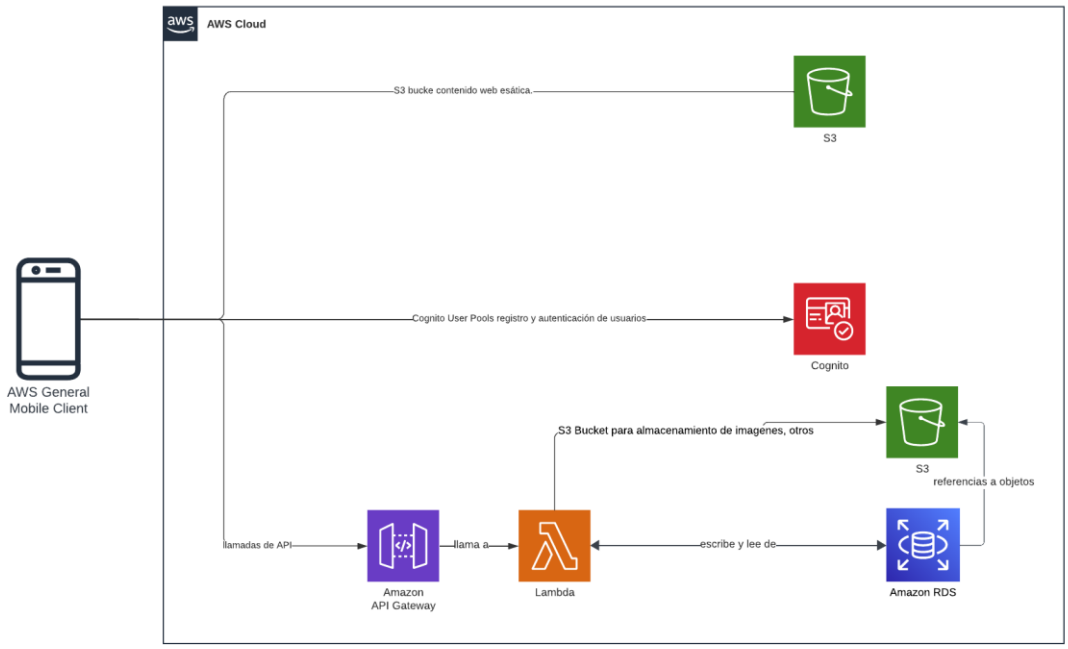


Figura 18. Vista de código del sistema
Fuente: Confeccionado por el autor

Como se puede observar en la Figura 18. *Vista de Código del Sistema* se encuentran los servicios a usar del proveedor de la nube y cómo se relacionan entre ellos. Esto ayuda a que la comunicación entre los miembros del equipo sea mejor a nivel técnico.

Los cuatro diagramas sobre el diseño de la aplicación del sistema permiten una vista completa de este, además que mejora la compresión al usar todos ellos. Juntos dan una vista holística - que era una consideración importante para seleccionar el lenguaje visual de modelado.

Principios y Decisiones de la Arquitectura

En esta parte se detalla cuál es el uso que se da a cada una de las plantillas creadas, además de entender las decisiones que fueron tomadas, a partir de las razones detrás de los diagramas presentados en la sección anterior.

En la Figura 19. *Principios Aplicación Experiencia Interactiva* se puede observar los principios creados por el equipo para la primera versión de este artefacto. Vale destacar cómo se deben tomar las decisiones sobre la arquitectura además las cosas que es necesario considerar al seleccionar los servicios involucrados en algunas de las decisiones.

El equipo de trabajo debe mantener en mente el alcance de la aplicación al tomar la decisión, a fin de basarla en un contexto adecuado y dando consideración a los objetivos de corto o mediano plazo.

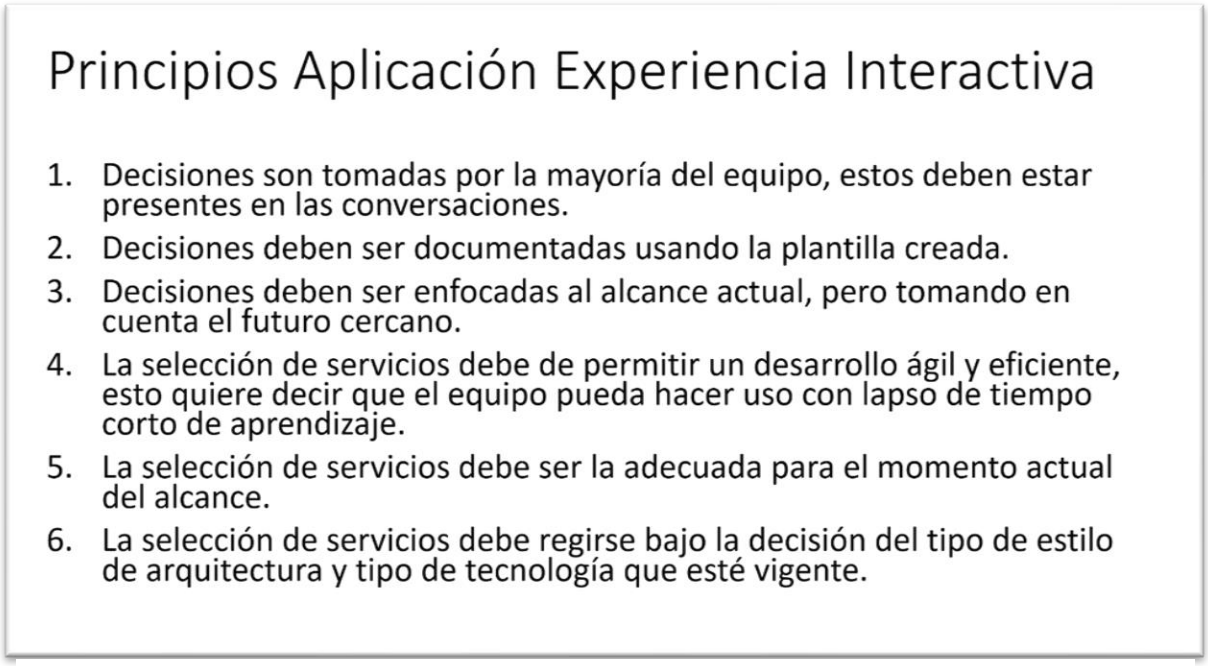


Figura 19. Principios aplicación experiencia interactiva
Fuente: Confeccionada por el autor

El segundo artefacto creado y utilizado por el equipo de trabajo son las decisiones tomadas para la creación de los diseños, estas pueden ser observadas en su totalidad en el Anexo: *Decisiones de Arquitectura*. A continuación, se detallan las razones de cada decisión sin entrar en un análisis profundo sobre ventajas o desventajas de su escogencia debido a que esto está fuera del alcance de esta investigación.

Las decisiones documentadas son para el alcance de la aplicación actual, que son relevantes para la investigación. Estas pueden ir variando con el tiempo debido al enfoque de arquitectura continua.

- **Proveedor de servicio de la nube:** la clave para tomar esta decisión fue el conocimiento interno que se tiene sobre este proveedor de servicio, debido a esto es que la balanza se inclina para hacer uso de los servicios de AWS.
- **Tipo de arquitectura de software:** el conocimiento interno vuelve a ser preponderante para tomar la decisión de usar un tipo de arquitectura en microservicios, no solamente por su flexibilidad sino también porque va a permitir un desarrollo ágil y con menos complicaciones debido a su naturaleza. Cabe destacar que la complejidad es dependiente al contexto y al cómo se implemente este tipo de arquitectura. Por ello, se recomendó seguir las buenas prácticas de la industria para no caer en anti-patrones.
- **Tipo de tecnología:** el equipo de trabajo no desea invertir tiempo en mantenimiento de servidores debido a que su capacidad no es la adecuada. Además, el alcance y enfoque de la aplicación debe priorizar funcionalidad sobre otras cosas, lo que puede ser atendido mediante la tecnología *serverless* y apoya el desarrollo ágil.
- **Framework de desarrollo móvil:** esta decisión fue la que más debate causó dentro del equipo, debido a que la pericia no es la más adecuada. Fue necesario realizar investigaciones para llegar a un acuerdo entre los miembros. Al final se seleccionó Angular-Ionic porque el *framework* cuenta con una estructura que es fácil de seguir para establecer el proyecto, pues permite que personas con poco conocimiento puedan hacer un desarrollo adecuado y conforme a las necesidades.
- **Base de datos:** la decisión sobre el servicio y la base de datos, contraria a la anterior, pues al analizar las opciones posibles, todo el equipo seleccionó RDS-MySQL, por

sus ventajas sobre otros servicios y por adecuarse de mejor manera a los objetivos del presente sistema de software.

Evaluación de Arquitectura

Una vez explicada la arquitectura, se procede a evaluarla con el objetivo de entender sus deficiencias e identificar mejoras posibles para realizar a futuro. Además, interesa recolectar información sobre la aplicación del lenguaje visual C4, a fin de determinar si cumple con el objetivo de comunicar lo deseado.

La evaluación seguirá un enfoque ligero debido al limitante tiempo y el alcance de la arquitectura presentada en la sección anterior. Se sigue el proceso expuesto por Nord et al (2009) para realizar una revisión estructurada de la documentación de una arquitectura, con base en 4 pilares o preguntas. En la Tabla 21. *Evaluación de la arquitectura de acuerdo con los pilares de Nord et al (2009) y su uso en la investigación* se puede observar los pilares, cuál es el concepto presentado por los autores y cómo es usado en esta investigación.

Pilar/Pregunta	Definición	Investigación
¿Por qué? - Propósito	Así como la arquitectura no es mala o buena, solo es adecuada o no para su uso, sucede lo mismo con la documentación. Ejemplos de por qué se realiza una evaluación: ● Conformidad con una normativa ● Idoneidad para el uso de la arquitectura para el propósito propuesto ● Idoneidad para evaluación o análisis formal	Para esta investigación se tienen 3 propósitos diferentes: ● Evaluar el uso de la notación C4 de acuerdo con la lista de verificación del sitio oficial de C4 ● Evaluar la notación usada con los principios de Moody (2009). ● Evaluar la idoneidad de la arquitectura de acuerdo con los principios de calidad ISO25010
¿Quién? - Involucrados	La documentación es usada por involucrados con algún objetivo en mente. Dentro de ellos, se pueden mencionar: arquitectos, diseñadores, clientes, probadores, entre otros	Debido a las limitantes de conocimiento que tiene el equipo de trabajo, se recurre a dos ingenieros de software con experiencia variada en el desarrollo de aplicaciones web, para que colaboren en la evaluación de la arquitectura.
¿Qué? - Alcance de revisión	Los documentos que serán revisados en la evaluación que	Los documentos por revisar solamente son: los diseños creados

	permitan cumplir con el propósito propuesto.	para la arquitectura de software de acuerdo con C4.
¿Cuándo? - Momento de revisión	De acuerdo con la metodología usada, así debería ser el momento o momentos donde se realiza la revisión de la arquitectura.	Debido a la limitante de tiempo y alcance de la investigación este tipo de revisión, sólo se realizará una única vez y con el propósito mencionado.

Tabla 21. Evaluación de la arquitectura de acuerdo con los pilares de Nord et al (2009) y su uso en la investigación
Fuente: Confeccionada por el autor

Proceso de la revisión

Luego de detallar el alcance de la revisión de la arquitectura se llevó a cabo una serie de pasos con el fin de obtener la información necesaria, vía análisis, conforme al propósito de la evaluación de la arquitectura.

En la Figura 20. *Proceso de la revisión de la arquitectura* se pueden observar los 4 pasos o fases las cuales se encuentra dividido el proceso ligero (liviano) de evaluación. Estas fases son descritas a continuación:

1. **Documentación del Contexto**

En esta fase se compartió un documento que fue creado para que los revisores pudieran entender el contexto del software: el objetivo, el uso y alcance de la información que van a revisar. Este documento aparece en el Apéndice 8, *Documentación del contexto*.

Dentro de la información que contiene dicho documento se encuentra lo siguiente:

- Definiciones relevantes tanto técnicas como del contexto sobre la industria de GodiTours.
- Relevancia del software.
- Flujo de la aplicación para el cual fue diseñada la arquitectura.
- Proceso de alto nivel de la investigación.
- Recomendaciones para realizar la revisión.

2. **Reunión del Kick-off**

Luego de que los revisores tuvieran un espacio para familiarizarse con el contexto del software, se procede a tener una reunión con ellos. El objetivo de la reunión fue detallar los siguientes puntos:

- Resolver dudas del contexto.
- Revisión de información general relevante al software.
- Explicación de los diagramas y diseños de arquitectura del software.

La presentación usada en esta reunión se puede ver completa en el Apéndice 8, *Presentación de la revisión de arquitectura*.

3. Evaluación offline de la propuesta

A los revisores se les da una semana para que, de manera *offline*, respondan al formulario que se les envió mediante un correo electrónico, luego de la reunión de lanzamiento. Se procedió de esta manera en atención a las limitaciones de tiempo de estas personas.

Como se mencionó en la Tabla 21. *Evaluación de la arquitectura de acuerdo con los pilares de Nord et al (2009) y su uso en la investigación*, el formulario está dividido en tres partes, de acuerdo con cada objetivo. Las partes se describen a continuación:

- **Notación C4:** en el sitio oficial de C4, se encuentra una lista de verificación sencilla que permite validar si los diseños realizados con este lenguaje cuentan con lo esencial. Con esta sección se desea comprender si se dio un buen uso general de los diagramas y la notación específica de este lenguaje.
- **Principios Moody:** La Tabla 13. *Principios para la eficacia cognitiva de la notación visual* está basada en Moody (2009). Esta sección busca obtener información sobre la notación utilizada de C4 y las personalizaciones que fueron realizadas. El objetivo es validar si se siguen estos principios o no, y determinar si es necesario realizar mejoras a futuro.
- **Evaluación Técnica/Tecnológica basada en ISO 25010:** con fin de determinar si la arquitectura cumple con los objetivos planteados por GodiTours en su primera versión, se selecciona este *framework* debido a su uso en la industria para evaluar omisiones, deficiencias o mejoras en las características principales.

El formulario completo puede ser revisado por completo en el Apéndice 9, *Formulario para la revisión de la arquitectura*.

4. Recolección de la información

Una vez completado el formulario por los revisores, se recolecta la información y realiza un análisis de los puntos que son relevantes de cada sección. La evaluación se hace en dos partes, una externa (sobre lo encontrado por los revisores) y una reflexiva (introspección personal sobre decisiones tomadas y que se pueden mejorar o modificar a futuro).

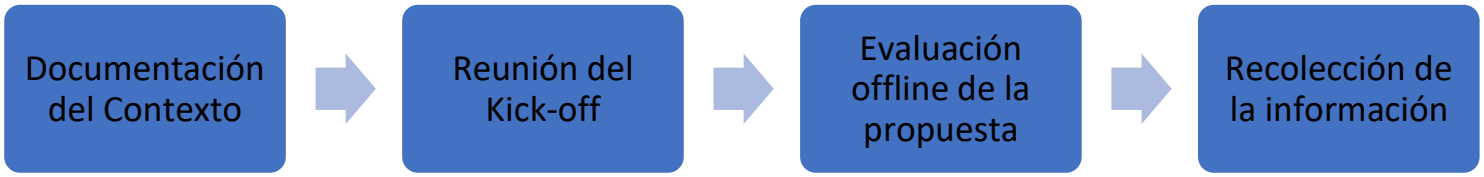


Figura 20. Proceso de la revisión de la arquitectura
Fuente: Confeccionada por el autor

El análisis se hace con base en la información brindada por los revisores. Se ha separado, acorde con las secciones del formulario. Así se brindan los puntos relevantes para cada sección. Si se desea ver a más detalle la información recolectada, esta aparece en el Apéndice 10, *Resultados del formulario para la revisión de la arquitectura*.

Evaluación externa

Notación C4

Como se puede ver en el Apéndice 9, todas las preguntas fueron respondidas por los revisores con un sí, lo que da a entender que incluso personas externas al grupo de trabajo que no estuvieron presentes en la creación de los diagramas comprenden la finalidad y funcionalidad de cada uno de estos.

Si bien había preocupación previa debido a los cambios realizados para adecuarse a las necesidades del equipo de trabajo, los resultados demuestran que estos no afectan la notación. La notación es mejorada porque permite una mejor compresión al dar más información.

El diagrama de código fue modificado respecto de la definición oficial. Esto no afectó la revisión y sigue cumpliendo con las buenas prácticas recomendadas oficialmente por C4.

A partir de lo anterior, se concluye que el uso de la notación de C4 cumple con las buenas prácticas de una manera satisfactoria, contribuyendo al entendimiento de los diversos involucrados o personas que hagan uso de este tipo de documentación.

Principios Moody

En esta sección se obtuvo resultados positivos. Se debe destacar 4 principios que tuvieron una calificación de medio a alto como se puede observar en la Figura 21. *Resultados de los principios según Moody (2009)* contra los diseños, por lo que a continuación se analiza la razón detrás de estos resultados - para cada principio.

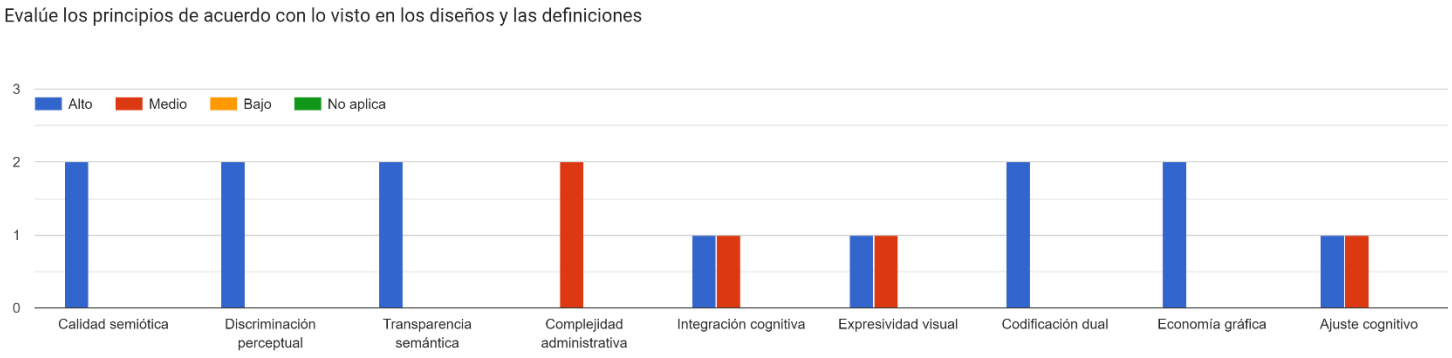


Figura 21. Resultados de los principios según Moody (2009) contra los diseños
Fuente: Confeccionada por el autor

- Complejidad administrativa: este principio concierne a los mecanismos que posee el modelo para ajustar la complejidad de acuerdo con la demanda. Como lenguaje, C4 permite reducir la complejidad al mantener una estructura de 4 diagramas que permitan ver el sistema de software de una manera adecuada. Sin embargo, C4 no incluye otro tipo de diagramas - como flujos de negocio - por lo que se entiende como una posible extensión futura, con diagramas que permitan solo ver lo que la audiencia necesita comprender o analizar. Para esta versión de la aplicación no representa un riesgo. No obstante, al crecer los servicios requerirse buscar alternativas para representarlos en un diagrama (de flujo de trabajo o de orquestación).

- Integración cognitiva: la definición de este principio es que, si existen varios modelos o diagramas, se deberían tener maneras de incluir la información de otros modelos en aras de lograr una búsqueda y entendimiento más sencillo. El software empleado para crear los diagramas no permite hacerlo; la integración entre diagramas se torna imposible. Hay software de pago que permite este tipo de funcionalidad, por su costo económico esto no es una opción para este proyecto.
- Expresividad visual: el principio establece que se debe usar un rango completo de variables visuales para aumentar la expresividad. En los diagramas creados, se usaron pocos colores para separar segmentos o para las relaciones, lo que explica la calificación dada en este aspecto. Esto puede ser considerado para mejoras futuras a los diagramas.
- Ajuste cognitivo: el modelo debe ofrecer diferentes representaciones de acuerdo con las circunstancias. C4 no tiene una manera de representar 'nativamente' flujos de negocio o diagramas entidad-relación, entre otros. En este caso no hay mayor problema, pero si se desea seguir aumentado servicios o funcionalidades, los diagramas deben ser extendidos.

De manera general se puede ver cómo las calificaciones de *medio* no afectan mucho el entendimiento del diagrama. Tomarlas en cuenta puede mejorar la comprensión, para beneficio de la audiencia.

Evaluación Técnica/Tecnológica basada en ISO 25010

Dentro de las 8 principales características de calidad de productos de software cubiertas por el estándar ISO 25010, las siguientes 4 fueron destacadas en los comentarios de los revisores:

- Eficiencia de desempeño: una mejora que se puede hacer en este aspecto es la evaluación de uso de algún tipo de cache para S3, esto si aumenta el uso de la aplicación y también de acuerdo con la cobertura y velocidad de acceso a Internet que posean los usuarios en algunos tours.
- Compatibilidad: el usar servicios propietarios de AWS, proveedor de servicio de la nube, puede provocar problemas si se desea cambiar a otro proveedor o plataforma.
- Usabilidad: debido a la cantidad de teléfonos celulares con Android, tomar en cuenta la variedad de tamaños de pantalla, para que eso no afecte a los usuarios.
- Fiabilidad: en esta característica una recomendación de los revisores es hacer uso de redundancia en el servicio de S3, caso de presentarse pérdidas de información por caídas del servicio o corrupción de los archivos, entre otros.

Los puntos mencionados son para mejoras que pueden ser atendidas cuando la aplicación ya se encuentre en una fase madura del proyecto y su uso sea más intenso. Por el alcance que se tiene en este momento, se está evitando utilizar más recursos en características que no se necesitan todavía. Es importante tener presentes las recomendaciones, a fin de atenderlas cuando la demanda de servicios crezca.

Evaluación reflexiva

Esta evaluación es una revisión personal sobre el diseño que fue creado por el equipo de trabajo, se trata de realizar una introspección de las decisiones tomadas para los diseños, a fin conceptualizar mejoras a futuro. Esta evaluación se hace de manera similar a la que los revisores realizaron, pero es más general y profundiza en los puntos que sea necesario.

Notación C4

Con la notación C4 los diseños se realizaron de acuerdo con las prácticas que son recomendadas en el sitio oficial. Por ello, al ser respondida toda la lista de verificación de manera afirmativa, se puede destacar que la personalización de la notación de acuerdo con el contexto colaboró mucho con su entendimiento por el equipo.

Además, el uso de diferentes niveles y el cambio del último nivel es una buena utilización de los diagramas que C4 permite, pues da flexibilidad para mostrar información que es necesaria para la implementación del software.

Principios Moody

Los principios que pueden ser mejorados en futuros diagramas o actualizaciones de estos, son como sigue:

- Calidad semiótica: se usaron diagramas en su mayoría con formas rectangulares para representar los diferentes servicios o aplicaciones que se encuentran dentro del software. Esto hace complicado de entender para la audiencia y le toma más tiempo de procesamiento; sería buena idea usar otras formas dependiendo de la funcionalidad o de alguna característica que haga sentido. Por ejemplo: usar formas de acuerdo con si se trata del *front-end*, las APIs o la base de datos.

- Discriminación perceptual: usar un solo tipo de forma provoca que se tome más tiempo para que la audiencia procese la información. Las formas diagramáticas que use el equipo de trabajo deben ajustarse conforme a la necesidad.
- Integración cognitiva: las limitaciones de recursos impidieron trabajar en la integración de los diferentes diagramas en uno solo, con el fin de que se pueda entender el sistema completo. Se recomienda que conforme vayan creciendo o aumentando los servicios o características del sistema se integren los diagramas progresivamente, cuidando que se controle la complejidad.
- Expresividad Visual: el uso de colores hubiera ayudado a reducir el tiempo de comprensión de los diagramas u otras variables que también permiten una adecuada recepción de la información.

Evaluación Técnica/Tecnológica basada en ISO 25010

Las oportunidades de mejora arquitectónicas a futuro se describen a continuación:

- Usabilidad: en esta característica, la accesibilidad fue un punto que no recibió suficiente atención debido al nicho de GodiTours y a los objetivos de la aplicación. La accesibilidad debe ser tomada en cuenta a futuro: una posible diversificación de servicios puede requerir su atención más allá de las configuraciones por defecto del dispositivo por usar.
- Fiabilidad: es una característica que tuvo poca preponderancia a la hora de realizar los diseños debido al alcance actual. Para un lanzamiento oficial o mejoras dentro del sistema debe tomarse en consideración. Por ser una aplicación donde el usuario puede presentar conexiones débiles de comunicación, son importantes la tolerancia a fallos y la capacidad de recuperación.
- Seguridad: este aspecto fue delegado completamente a la configuración de los servicios del proveedor de la nube. Si se desea manejar pagos u otros servicios se recomienda cuidar este aspecto para no provocar pérdida de datos importantes o incluso credibilidad ante los usuarios.
- Portabilidad: la tecnología usada en este producto de software crea cierta dependencia al usar servicios de un proveedor de nube. Se intentó tomar decisiones que fueran agnósticas en lo posible para, en caso de necesitar mudarse a otro proveedor, reducir el esfuerzo requerido para migrar y levantar servicios nuevos. Por ello, a futuro se recomienda poner mucha atención a los requerimientos de portabilidad cuando se elijan servicios.

La investigación ha permitido trabajar metódicamente mediante una combinación de procesos y herramientas que facilitan la toma de decisiones y la expresión del diseño. El lenguaje visual de modelado fue seleccionado racionalmente y su usabilidad ha sido estudiada para determinar la productividad en las conversaciones de requerimientos, conceptualización y arquitectura, así como en la futura implementación del software.

Lo obtenido en este proyecto puede contrastarse con experiencias laborales previas. Al implementar ciertas prácticas se causó un impacto positivo en cuanto al valor y la eficacia con la que se trabaja. En la Tabla 22. *Comparación organizacional*, se detallan las características que permiten comparar las formas de trabajo anteriores con las que este proyecto deja en la organización: mayor claridad en las conversaciones con los involucrados del negocio, mejor comprensión del contexto, abordaje adecuado de los problemas para darles solución.

Característica	Pasado	Presente
Toma de decisiones	Individual	Equipo
Documentación de decisiones	Nula o desactualizada	Estándar con formato ágil
Diseños de Arquitectura	Sin formato específico	Formato elegido por el equipo
Principios por seguir	No existía	Referencia de arquitectura organizacional

Tabla 22. Comparación organizacional
Fuente: Confeccionada por el autor

Las herramientas desarrolladas en este proyecto, junto con el proceso implementado, ofrecen una base sólida para la organización y los objetivos que ellos desean implementar. Se usan prácticas comprobadas en la industria, adaptadas al contexto para que la adopción sea orgánica y no invasiva para el equipo de trabajo.

Crítica Lenguaje Visuales de Modelado

Como se ha explicado, los lenguajes visuales de modelado tienen cerca de 5 décadas de estar presentes en la industria. A pesar de que existen cursos formales e informales y muchos libros (particularmente para UML), en su investigación Baltes y Diehl (2015) encontraron que los diagramas y bocetos sin seguir ninguna sintaxis o regla especial es lo más usado en la industria de la Ingeniería del Software, a pesar de que la eficiencia y efectividad en la comunicación se vea reducida por esto.

Es interesante observar que los lenguajes visuales continúan evolucionando y actualmente se encuentran muchas variaciones de estos, lo que evidencia que ningún lenguaje visual cumple a cabalidad con lo que se necesita en la industria. Los usuarios de los lenguajes visuales terminan combinándolos y adaptándolos, para contar con las características que habiliten una comunicación adecuada entre los diferentes involucrados en los proyectos.

En la industria es un problema generalizado brindar soluciones, sin comprender el usuario o su real necesidad. Al parecer, la creación de nuevos lenguajes visuales de modelado se produce pensando en puntos de vista limitados, lo que consecuentemente es difícil de generalizar para la industria.

La investigación realizada por Moody (2009), a pesar de no contar con un soporte amplio de la industria y academia, ha tenido apoyo en la parte científica: Ziehmann y Lantow (2021) mencionan que ya para entonces se tenían más de 700 citaciones. Muchos artículos han usado los principios de Moody para realizar evaluaciones, de manera similar a lo reportado en la presente investigación. Nuestro trabajo podría ser usado en la industria para verificar y validar si los lenguajes visuales de modelado cumplen con una comunicación adecuada. Además, las buenas prácticas que se recomiendan pueden ser incluidas en herramientas para mejorar los diagramas que se crean, desde bocetos en adelante.

La experiencia con GodiTours en la creación de la arquitectura para el software de una experiencia interactiva permitió usar C4 como lenguaje visual, después de haberlo seleccionado metódicamente. Es posible plantear una crítica a C4 con base en los principios de Moody, mediante una revisión semi-formal del lenguaje como tal, gracias a la experiencia ganada mediante su aplicación en un contexto específico.

Crítica C4

Calidad Semiótica

C4 no cuenta con una sintaxis específica, pues los usuarios pueden elegir sintaxis a voluntad. El sitio web oficial ofrece ejemplos con una notación específica que también fue detallada en la Tabla 20. *Notación comparativa C4 y GodiTours*. Dicha notación falla en los siguientes puntos:

- Sobrecarga de símbolos: C4 representa prácticamente todo con rectángulo. Se usa un símbolo para muchos significados. Esto provoca dificultad de lectura y entendimiento para la audiencia.
- Déficit de símbolos: en la Ingeniería del Software se tiene muchos conceptos, procesos, servicios y sistemas, entre otras cosas, C4 ofrece pocos símbolos y eso provoca que la diferenciación sea complicada y se esté afectando la eficiencia de los diagramas.

A pesar de lo indicado, en C4 se permite usar otras variables (Principio de Expresividad Visual), que permiten identificar el símbolo, junto con la funcionalidad del objeto que se está representando.

Discriminación Perceptual

En este principio, C4 hace un buen uso de las variables visuales agregando texto tanto en contenedores como en las relaciones, lo cual ayuda a entender el mensaje que se desea transmitir y que, aunque los símbolos tengan formas similares se distinga la funcionalidad de cada uno.

Transparencia Semántica

La inferencia del significado de cada símbolo en la notación de C4 puede ser compleja. Por ejemplo, se añadieron iconos en los diagramas de GodiTours para que la audiencia reconozca rápidamente su significado. Esto podría ser algo por mejorar en la notación de C4.

Conforme a este principio, en esta notación las relaciones cuya compresión es natural para el receptor: cada relación tiene un significado, descrito por la acción y dirección de esta - una relación va con *una* acción y no muchas acciones con una relación como se da en otros lenguajes.

Complejidad Administrativa

C4 posee una representación escalable, sus 4 diagramas van adentrándose en cada nivel del sistema, lo que facilita comprender su composición.

Lo que puede ser un problema es la cantidad de información representada en un diagrama. En el sitio oficial de C4 no se da información sobre esto.

La integración de los diagramas se puede realizar de diferentes maneras. Existen herramientas de software que apoyan tal integración, pero pueden estar fuera del alcance

presupuestario de muchas organizaciones. Es necesario buscar maneras de realizar la integración; sin herramientas se depende de la disciplina que tenga el usuario.

Integración Cognitiva

En el estado del arte actual, una integración sencilla de los 4 diagramas de C4 solo se puede lograr mediante software de pago y con disciplina.

Expresividad Visual

Los lenguajes visuales de modelado investigados, incluido C4, no aprovechan suficientemente las variables visuales. Entre las variables visuales cabe destacar:

- Color: en el sitio oficial de C4 se usan colores, pero no para diferenciar los símbolos y que esto ayude a comprender mejor el significado.
- Forma: la mayoría de los símbolos de C4 son rectángulos, lo que dificulta una distinción adecuada y exige acudir al texto para poder comprender el significado.
- Posición: las recomendaciones de C4 no brindan instrucciones o guías sobre cómo deben ser posicionados los símbolos en el diagrama para que la lectura sea natural y sencilla para la audiencia.

Aunque haya otras variables visuales, debe cuidarse la accesibilidad dentro de los diagramas con el fin de que la audiencia no se vea afectada por eso.

Codificación Dual

Este principio es uno de los cuales cumple a cabalidad C4, debido a que usa texto para complementar los símbolos y las relaciones; asimismo, permite describir brevemente la funcionalidad de cada contenedor y parte del sistema.

También puede agregarse la tecnología como parte del texto que se coloca en cada símbolo, lo cual favorece el entendimiento y reduce la carga cognitiva (memoria de trabajo) conforme se van analizando los diagramas.

Economía Gráfica

C4 tiene pocos símbolos que permiten cumplir este principio, pero crea tensiones cono los dos primeros principios, por lo que se debe buscar un punto medio que permita cumplir con estos. Para ello, añadir iconos ayuda a que C4 cumpla con más principios.

Ajuste Cognitivo

La división en cuatro diagramas que plantea C4 permite que estos sean enfocados a diferentes audiencias. Ciertos diagramas son enfocados técnicamente por lo que puede depender de cómo se creen los diagramas y llegar a romper otros principios.

Resumen

Como se puede ver en la Tabla 22. *Evaluación de C4 contra los principios según Moody (2009)*, C4 logra tener una nota regular si comparamos todos los principios. Esto se debe a la flexibilidad de este lenguaje, que permite a sus usuarios modificar la notación que se incorpora en los diagramas.

Principio	**Evaluación de Cumplimiento**
Calidad Semiótica	Bajo
Discriminación Perceptual	Bajo
Transparencia Semántica	Medio
Complejidad Administrativa	Bajo
Integración Cognitiva	Medio
Expresividad Visual	Bajo
Codificación Dual	Alto
Economía Gráfica	Alto
Ajuste Cognitivo	Medio

Tabla 23. Evaluación de C4 contra los principios según Moody (2009)
Fuente: Confeccionada por el autor

La ventaja en flexibilidad del lenguaje genera dependencias en ciertos principios. Quienes creen diagramas o diseños son responsables de seguir buenas prácticas - encaminadas a lograr buena comunicación: La industria debe aprender técnicas para mejorar la comunicación de información: el objetivo es que el mensaje transmitido coincida con el recibido por la audiencia.

Conclusiones

El objetivo principal de esta investigación era evaluar una herramienta para seleccionar un lenguaje de modelado para el diseño de una arquitectura para la aplicación de software que desea desarrollar GodiTours, debido a la importancia que tienen las expresiones de diseño y - particularmente - los diagramas en el ámbito técnico, así como en las conversaciones del negocio. La investigación debió considerar los recursos limitados de la empresa, tanto de conocimiento como de personal.

Primeramente, se investigaron los lenguajes de modelado usados en la industria con el fin de tener criterio teórico para poder comprender sus ventajas y desventajas, así como encontrar similitudes y diferencias que permitieron entenderlos mejor. Con ello se posibilita realizar comparaciones objetivas.

Posteriormente se realizó un estudio con expertos para poder comprender de primera mano cuál es el proceso de decisión por el cual han pasado al elegir entre la gran lista de lenguajes de modelado disponible para la industria. Asimismo, se estudió el contexto de la organización y sus asuntos relevantes.

Se realizó un análisis de la información recabada para obtener la lista de variables que formarían parte de la herramienta para seleccionar lenguajes de modelado. Lo anterior y los principios de Moody (2009) dieron fundamento al diseño de una herramienta que apoye la selección del lenguaje de modelado, conforme a las variables preponderantes para el contexto del software y de la organización.

Al finalizar esta etapa, se aplicó la herramienta junto con los miembros del equipo, que condujo a seleccionar a C4 para el caso de GodiTours. Además, se obtuvo valoraciones positivas de la herramienta y el proceso que se usó, gracias a su facilidad de uso y las discusiones que llevaron a un consenso grupal.

Por último, para poder realizar una crítica adecuada al lenguaje seleccionado, se diseñó una arquitectura de software para la aplicación de GodiTours. La arquitectura fue sometida a evaluación por expertos, considerando tres aristas: las buenas prácticas propias de C4, los principios de Moody (2009) y el estándar ISO 25010 como marco de calidad de productos de software. Así, fue posible

realizar un análisis profundo del lenguaje de modelado “en acción” y determinar puntos de mejora para este.

Al concluir este proyecto se logra cumplir el objetivo principal y los objetivos específicos, así como diseñar una arquitectura de software con la cual GodiTours inició el desarrollo de su deseada aplicación. El diseño de la arquitectura, expresado en el lenguaje de modelado C4 extendido con iconos, ha dado fluidez a las conversaciones entre los niveles técnicos y de negocio que se dan en un proyecto como estos. Además, ahora se cuenta con más información para mejorar la notación C4 en uso, así como del diseño gracias a las observaciones realizadas durante el proceso.

De forma general, los productos de este proceso cumplieron con lo planteado desde el inicio, además se facultó a la organización para hacer mejoras al diseño autónomamente con base en sus propias decisiones. Este proyecto también confirma que en la Ingeniería del Software el contexto es lo más relevante al tomar decisiones, pues depende de la industria, dominio o necesidad que motiva las iniciativas y donde deben enmarcarse las soluciones.

Recomendaciones

Las recomendaciones son enfocadas en tres áreas diferentes:

- GodiTours: alcance organizacional y la aplicación por desarrollar.
- Academia: organizaciones o personas que de manera formal o informal compartan conocimiento en la Ingeniería del Software.
- Industria: enfocada a la Ingeniería del Software.

GodiTours

La organización debe apuntar a elaborar prototipos de una manera eficiente para obtener retroalimentación rápidamente y desarrollar una aplicación deseable para sus usuarios. Además, debe enfocarse en realizar las mejoras en la arquitectura mencionadas en este documento, con el fin de enriquecer la experiencia de usuario y simplificar el mantenimiento futuro del software.

La notación de los diagramas debe mantenerse clara para que los involucrados se comuniquen productivamente. Es necesario expresar en diagramas los procesos de negocios y alinearlos con la arquitectura; esto permitiría al equipo de trabajo comprender mejor los componentes involucrados en cada flujo de usuario.

Fuera del alcance de esta investigación, es necesario continuar elevando el conocimiento del personal de GodiTours, para que a futuro puedan atender nuevos problemas que surjan en la evolución futura de la aplicación.

Academia

Opino que parte de la problemática que se ha presentado en la Ingeniería del Software es intentar basarse o copiar lo que otras ingenierías hacen. Las ingenierías 'clásicas' desarrollan prácticas que buscan soluciones que pueden ser usadas en la mayor parte de una industria o área de actividad. Esto es muy problemático en el área de software, pues el *contexto* es sumamente importante al desarrollarlo. En la educación hace falta estudiar más los contextos: los espacios de problemas, los dominios de aplicación, las restricciones organizativo-sociales o físicas, las regulaciones técnicas, las familias de tecnologías, los marcos legales y éticos, e incluso las culturas que encuadran el desarrollo, el mantenimiento y la evolución del software. Se debería enfocar los esfuerzos en enseñar principios (generales) en lugar de soluciones (específicas) a fin de conseguir flexibilidad para comprender el contexto y concebir soluciones pertinentes a los problemas que interesan a los usuarios.

En tiempos recientes han tomado mayor relevancia las habilidades blandas, también llamadas socioemocionales. Por ejemplo, en esta investigación se ha mencionado mucho la importancia de los diagramas para lograr una comunicación efectiva entre los involucrados y evitar contratiempos, dirigiendo los esfuerzos al desarrollo de productos o servicios de software que den alta satisfacción a los usuarios. La academia tiene el reto de idear maneras en que los estudiantes desarrollen las habilidades blandas mientras cursen sus planes de estudio - con el fin mejorar esta área de los futuros ingenieros.

Industria

Algo que ha aquejado a la Ingeniería del Software, tanto externamente como internamente, es brindar soluciones que se creen viables debido a suposiciones o creencias personales. Esto ha provocado que tales soluciones se repliquen y fallen en múltiples implementaciones, pues se olvida que el contexto importa. En lugar de desarrollar soluciones descontextualizadas, se debería dilucidar cuáles fueron las razones por las que una solución funcionó en su contexto original y qué principios pueden ser aplicados en otros contextos.

Otro problema que se presenta al aplicar tales soluciones es presentar ideas o características no esenciales, que al final el usuario no desea, lo cual induce un uso que disminuye con el tiempo. Como ejemplo podemos ver la cantidad de lenguajes visuales de modelado que se han presentado a lo largo de la historia: ninguno de ellos ha sido acogido por la industria, pues no se ha podido entender qué desean los *usuarios* (ingenieros o no); como ingenieros no hemos comprendido bien el problema de la *usabilidad* de los lenguajes de modelado (visuales, textuales o combinados). Considero que pensar en *principios* ofrece una solución, pues brinda adaptabilidad al contexto y evita desarrollar sistemas de software que se abandonarán con el paso del tiempo.

Referencias Bibliográficas

- Ave Coders. (2021, 16 abril). All UML Diagrams in 10 minutes [Vídeo]. YouTube. https://www.youtube.com/watch?v=hF4yg1yFrdU
- Baltes, S., & Diehl, S. (2015). Sketches and Diagrams in Practice. Software Engineering & Management, 69-70. https://dblp.uni-trier.de/db/conf/se/se2015.html#BaltesD15
- Brown (2019, 16 agosto). Visualizing software architecture with the C4 model. Agile on the Beach 2019 [Video]. YouTube. Recuperado 13 de junio de 2023, de https://www.youtube.com/watch?v=x2-rSnhpw0g
- Brown, S. (s. f.). The C4 model for visualizing software architecture. C4 Model. Recuperado 13 de junio de 2023, de https://c4model.com/#Notation
- Cherubini, M., Venolia, G., DeLine, R., & Ko, A. J. (2007). Let's go to the whiteboard. https://doi.org/10.1145/1240624.1240714
- E. Woods and R. Hilliard, "Architecture Description Languages in Practice Session Report," 5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05), Pittsburgh, PA, USA, 2005, pp. 243-246, doi: 10.1109/WICSA.2005.15.
- Erder, M., & Pureur, P. (2015). Continuous architecture: Sustainable Architecture in an Agile and Cloud-Centric World. Morgan Kaufmann.
- Guthrie, G. (2021, 6 agosto). What is an architecture diagram, and why do you need one? | Nulab. Nulab. Recuperado 21 de abril de 2023, de https://nulab.com/learn/software-development/what-is-an-architecture-diagram-and-why-do-you-need-one

- ICT. (2022). LLEGADAS INTERNACIONALES DE TURISTAS POR PAÍS Y PUESTO MIGRATORIO [Conjunto de Datos]. https://www.ict.go.cr/es/documentos-institucionales/estad%C3%ADsticas/informes-estad%C3%ADsticos/recientes/2253-2022-3.html
- ISO 25010. (s. f.). Recuperado 15 de octubre de 2023, de https://iso25000.com/index.php/normas-iso-25000/iso-25010
- Josey, A. et al (2019). ArchiMate 3.1 Specification A pocket guide (4.a ed.). Van Haren.
- Kitch, B. (2023, September 7). How to create a team working agreement. Mural. Recuperado el 26 de setiembre de 2023, de https://www.mural.co/blog/team-agreement-guide
- Koschmider, A., Drescher, A., & Lehner, J. (2018). A Survey-based Analysis of Principles to Evaluate Visual Notations of Process Modeling Languages.
- Kruchten, Philippe. (1995). The 4+1 View Model of Architecture. IEEE Software. 12. 45-50. 10.1109/52.469759.
- Lucidchart (2019). UML Diagram Templates and Examples. https://www.lucidchart.com/blog/uml-diagram-templates
- Maier, M. W., Emery, D., & Hilliard, R. (2001b). Software architecture: introducing IEEE Standard 1471. IEEE Computer, 34(4), 107-109. https://doi.org/10.1109/2.917550
- Moody, D. (2009). The “Physics” of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. IEEE Transactions on Software Engineering, 35(6), 756-779. https://doi.org/10.1109/tse.2009.67
- Nord, R., Clements, P., Emery, D., & Hilliard, R. (2009, December 1). A Structured Approach for Reviewing Architecture Documentation. (Technical Note CMU/SEI-2009-TN-030). Retrieved November 27, 2023, from https://doi.org/10.1184/R1/6571757.v1.
- Object Management Group. (2011). BPMN 2.0. OMG. Recuperado 3 de abril de 2023, de http://www.omg.org/spec/BPMN/2.0
- Oficina Económica y Comercial de España en Panamá. (2021). Informe Económico y Comercial: Costa Rica. https://www.mapa.gob.es/es/ministerio/ministerio-exterior/america-central-caribe/ofecom-informeeconcomercial_may2021_tcm30-542367.pdf

- Salary: Software Engineer in Costa Rica May 2023. (s. f.). Glassdoor. Recuperado 24 de mayo de 2023, de https://www.glassdoor.com/Salaries/costa-rica-software-engineer-salary-SRCH_IL.0,10_IN57_KO11,28.htm
- Schwaber, K y Sutherland, J. (2020). La Guía de Scrum. Estados Unidos: Scrum.org
- Sommerville, I. (2015). Software Engineering (10a ed.). Pearson.
- Tang, A., van Vliet, H. (2009). Software Architecture Design Reasoning. In: Ali Babar, M., Dingsøyr, T., Lago, P., van Vliet, H. (eds) Software Architecture Knowledge Management. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-02374-3_9
- Taylor, R. J. K., & Van Der Hoek, A. (2007). Software Design and Architecture The once and future focus of software engineering. https://doi.org/10.1109/fose.2007.21
- The C4 model for visualising software architecture. (s. f.). Recuperado 8 de agosto de 2023, de https://c4model.com/#Tooling
- The European market potential for nature and ecotourism | CBI. (2020, 8 enero). Recuperado 22 de mayo de 2023, de https://www.cbi.eu/market-information/tourism/nature-tourism/nature-eco-tourism/market-potential
- The Open Group. (s. f.). Introduction: ArchiMate® 3.2 Specification. ©The Open Group, 2012-2023. Recuperado 13 de junio de 2023, de https://pubs.opengroup.org/architecture/archimate3-doc/ch-Introduction.html
- Tsenov, M. (2022, June 28). The Importance of Software Architecture. Dreamix Group. Recuperado 3 de junio de 2023, de https://dreamix.eu/blog/frontpage/the-importance-of-software-architecture
- Unir, V. (2022, 2 junio). ¿Qué es el lenguaje visual? Elementos, tipos y ejemplos en comunicación. UNIR. Recuperado 17 de abril de 2023, de https://www.unir.net/ingenieria/revista/lenguaje-visual/
- Van Heesch, U., Avgeriou, P., & Hilliard, R. (2012). Forces on Architecture Decisions - A Viewpoint. Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture, Helsinki, Finland, 2012, pp. 101-110. https://doi.org/10.1109/wicsa-ecsa.212.18
- Victor. (2023). 7 modelling languages for software architecture. Ice-website. Recuperado 14 de junio de 2023, de https://icepanel.io/blog/2023-02-28-7-modelling-languages-for-software-architecture

- What is Architecture Diagramming? - Software & System Architecture Diagramming Explained - AWS. (s. f.). Amazon Web Services, Inc. Recuperado 23 de mayo de 2023, de https://aws.amazon.com/what-is/architecture-diagramming/
- What is SDLC? - Software Development Lifecycle Explained - AWS. (s. f.). Amazon Web Services, Inc. Recuperado 21 de abril de 2023, de https://aws.amazon.com/what-is/sdlc/#:~:text=The%20software%20development%20lifecycle%20(SDLC,expectations%20during%20production%20and%20beyond.Balosin, I. (2019, 19 enero). Why Do We Need Architectural Diagrams? InfoQ. https://www.infoq.com/articles/why-architectural-diagrams/
- What is UML? Everything You Need to Know About Unified Modeling Language. (2020). Gliffy by Perforce. https://www.gliffy.com/blog/what-is-uml-everything-you-need-to-know-about-unified-modeling-language#:~:text=implements%20the%20next.-,History%20of%20the%20Unified%20Modeling%20Language,%2C%20IBM%2C%20and%20Apple%20Computer.
- Woods, E. (2023, 14 septiembre). Democratising Software Architecture • Eoin Woods • GOTO 2023 [Vídeo]. YouTube. https://www.youtube.com/watch?v=nchRmYvUf2Y
- Ziehmann, J., & Lantow. (2021). Moody's Physics of Notation: High Impact, Little support. CEUR-WS, 3045(04), https://ceur-ws.org/Vol-3045. https://ceur-ws.org/Vol-3045/paper04.pdf

Apéndices

1. Cuestionario a expertos

Elección de un lenguaje visual de modelado para el diseño de una arquitectura de software

Autor: Juan González Núñez
Población: Expertos en Ingeniería de Software
Objetivo: Evaluar cualitativamente el proceso en la toma de decisiones y el uso que se le da a los lenguajes visuales de modelado en el desarrollo de software.
Estructura: Esta herramienta posee preguntas con el fin de entender cómo se elige un lenguaje visual de modelado para el diseño de una arquitectura de software.

jgonzalezn@ucenfotec.ac.cr Cambiar de cuenta

No compartido

¿Cuáles lenguajes visuales de modelado que conoce?

Tu respuesta

¿Cúal es el proceso para elegir el lenguaje visual de modelado a la hora de realizar un diseño?

Tu respuesta

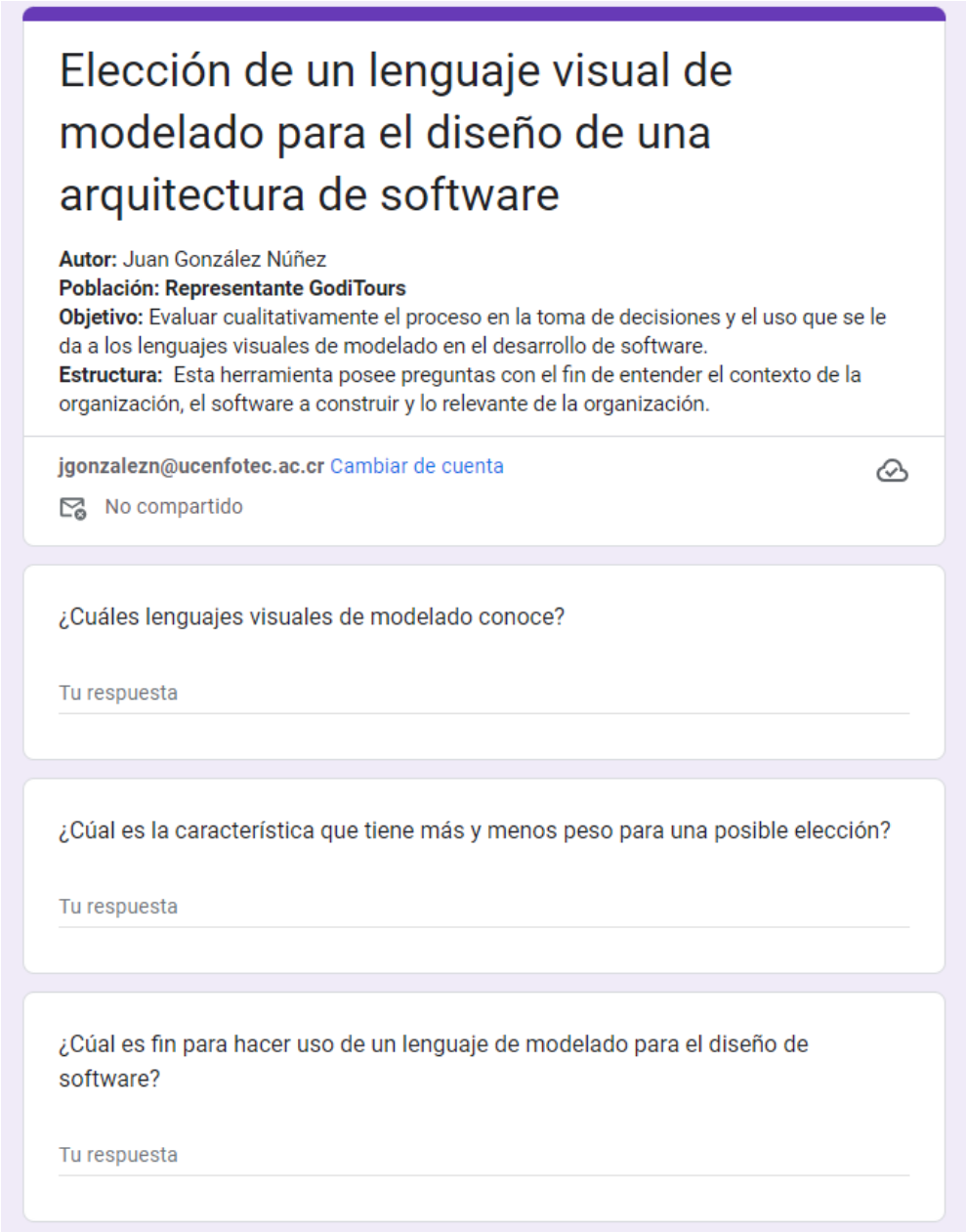
¿Cúal es la característica que tiene más y menos peso para la elección?

Tu respuesta

¿Cúal es fin para hacer uso de un lenguaje de modelado para el diseño de software?

Tu respuesta

2. Cuestionario GodiTours



Elección de un lenguaje visual de modelado para el diseño de una arquitectura de software

Autor: Juan González Núñez
Población: Representante GodiTours
Objetivo: Evaluar cualitativamente el proceso en la toma de decisiones y el uso que se le da a los lenguajes visuales de modelado en el desarrollo de software.
Estructura: Esta herramienta posee preguntas con el fin de entender el contexto de la organización, el software a construir y lo relevante de la organización.

jgonzalezn@ucenfotec.ac.cr Cambiar de cuenta

No compartido

¿Cuáles lenguajes visuales de modelado conoce?

Tu respuesta

¿Cúal es la característica que tiene más y menos peso para una posible elección?

Tu respuesta

¿Cúal es fin para hacer uso de un lenguaje de modelado para el diseño de software?

Tu respuesta

3. Información cuestionario a expertos

Población	¿Cuáles lenguajes visuales de modelado que conoce?	¿Cuál es el proceso para elegir el lenguaje visual de modelado a la hora de realizar un diseño?	¿Cuál es la característica que tiene más y menos peso para la elección?	¿Cuál es el fin para hacer uso de un lenguaje de modelado para el diseño de software?
Administrador de Producto	UML	En la experiencia basada no se le da tanta importancia al lenguaje, si no más se ha usado no un lenguaje en específico se usa un entendimiento grupal sin usar un lenguaje en específico. En ciertos diagramas los arquitectos ya vienen con la decisión del lenguaje a usar y es presentado. No hay una forma de decir lo que se use un estándar u otro.	Mayor peso: -Representación de vistas: Niveles y profundidad. -Flexibilidad: Componentes a la medida, personalización. Ejemplo: SaaS platform. CNCF Landscape Menor Peso: -No hay relevancia entre pasarlo a diagrama a código. -No importa si hay un ente detrás certificando el estándar.	Usamos en conversaciones planeamiento, con involucrados para toma de decisiones, resolución de problemas, creación de backlog para ver los puntos de fricción. Estado futuro.
Arquitecto	UML - Diagramas de Flujo	Influye mucho el sesgo de la persona, quiere decir a que tipo de experiencia tengo. Empiezo con UML a alto nivel.	Mayor Peso: - Complejidad - Soporte de la comunidad: para resolver preguntas, puede ser fácil de aprender, pero ante problemas específicos o escenarios que no sepa diagramarlo. Menor Peso: - Generación de código No importa el ente que lo creo o si es certificado.	Para todo tipo de conversación, por eso la facilidad es relevante porque así se puede explicar a personas técnicas o no técnicas para ejemplificar cosas. Es usado en las diferentes fases del SDLC, es como unos rayos X de la aplicación.

Desarrollador	UML, Entidad - Relación bases de datos.	Dependiendo del contexto y sobre las tareas solicitadas se toma la decisión. Si usase un lenguaje estandarizado o se quedaría con bocetos. Esta alejado de la realidad que en la empresa se use uno como estándar.	Mayor Peso: -Sencillo de aprender: entre más rápido sea el trasmitir la información entre las partes es mejor. Flexibilidad: que se pueda compartir, que se agreguen componentes personalizados. Menor Peso: -Estandarizado: no interesa el estándar a nivel de industria, me interesa pasar el conocimiento.	No uso diagramas para las conversaciones entre los miembros del equipo. Es rara vez como para ver flujos, pero mínimo.
Arquitecto	UML, C4	Depende del contexto del software y del objetivo del diagrama. Por ejemplo, he usado C4 que me permite tener conversaciones con mis líderes e ingenieros para las conversaciones. UML es complicado debido al conocimiento que se requiere. Lo más importante es que se entienda C4 permite tener el contexto, pero no te da la interacción total entre componentes o las secuencia de eventos.	Mayor Peso: - Compresión del lenguaje: intrínsicamente todos los modelos son erróneos. Scope de cada diagrama se entienda a la audiencia adecuada y que sea accesible. "Agnóstico". - Simple, curva de aprendizaje: que sin saber nada pueda entender y actualizar relativamente fácil. Menor Peso: - Estándares muy específicos: que sea muy detallado, entre más sencillo mejor. Mucho nivel de detalle no me interesa. Especificidad. Generación de código: a veces es contraproducente debido al retrabajo	Broad comunicación, blue prints que llevan una historia como un versioning para entender toma de decisiones. Lo más importante del diagrama es para comunicar.

Administrador de Producto	BPMN	Se bocetean las cosas no hemos usado un proceso formal de elección. No he sido parte activa de generar los diagramas solo como audiencia.	Mayor Peso: -Holístico: niveles o tipos de diagramas, de acuerdo con la audiencia que se pida. Que todos veamos a lo mismo y ahorre tiempo. Menor Peso: -Detalle técnico propiamente: ver información que necesito sin que me estorbe la otra vista. -Generalidad: ser lenguajes tan generales que ocupemos otra documentación para comprender.	Conversaciones con los involucrados para toma de decisiones; herramientas para conversaciones de timelines, priorización, mantenimiento, vender la solución a los involucrados sobre toma de decisiones. (negociaciones). Entendimiento del producto debido a lo que pasa por debajo.
Desarrollador	UML	No he usado uno, solamente bocetos.	Mayor Peso: -Holístico: que sea fácil de usar, nada muy complejo pero que sirva para conversar en diferentes perspectivas. - Flexible: que pueda usar gráficas que el equipo decida Menor Peso: -Estándar: no me interesa que sea uno	Conversaciones con diferentes tipos de personas, y que la toma de decisiones se vuelva sencilla debido al lenguaje todos entendemos lo que hablamos

4. Información cuestionario a GodiTours

¿Cuáles lenguajes visuales de modelado conoce?	¿Cúal es la característica que tiene más y menos peso para una posible elección?	¿Cuál es el fin para hacer uso de un lenguaje de modelado para el diseño de software?
UML, pero es conocimiento teórico, nunca se ha implementado nada	La verdad para nosotros la comunicación es muy relevante, por lo que poder tener diferentes aristas representadas es lo ideal; porque tenemos conversaciones tanto con gente que conoce la parte técnica como los que no. Y lo que ya hemos hablado el aprendizaje tiene que ser corto, no podemos invertir tiempo en aprender algo, porque tenemos que empezar a desarrollar lo más pronto posible. Lo que no nos interesa mucho es la generación de código, esto nos puede provocar demasiado retrabajo en un futuro.	Como dijimos anteriormente la comunicación con diferentes áreas, además que ocupamos que nos ayuden a tomar decisiones de una manera adecuada y efectiva.

5. Información entrevista post-implementación

¿Cuál es la retroalimentación con respecto de la documentación brindada?	¿Cómo fueron las discusiones que se tuvieron en las diferentes reuniones?	¿Cuál es su opinión con respecto de la herramienta y el proceso?
La información que fue compartida antes de las reuniones me ayudó a entender mejor la herramienta y el significado de las variables, no tenía muchas preguntas a la hora de que nos reunimos a usarla.	Las reuniones la verdad me gustaron por que pude observar los puntos de vista de mis compañeros y entender mejor su manera de pensar en ciertos aspectos.	La herramienta en sí es super sencilla la verdad, siento que las variables que teníamos que evaluar eran las que necesitamos en este momento, quizás en un futuro pueden variar, pero actualmente es lo que ocupa el proyecto. El proceso la verdad pensé que iba a ser más tedioso, pero fue sencillo. Recomendaría que la tabla fuera como un estilo formulario en la web, creo que sería menos complicado usarla.
Los diferentes archivos que se compartieron para entender cómo usar la herramienta son adecuadas, la verdad me fue sencillo usarla y entender cómo funcionaba.	Las discusiones fueron relevantes, ya que nos permitía entender no solamente lo que cada persona piensa, si no, también que es lo que se desea para el proyecto. Siento que esas reuniones nos ayudaron a sentar ciertas bases dentro del equipo para entendernos mejor.	La herramienta en sí es sencilla, sin embargo, para mejorar el uso del tiempo en las reuniones sería interesante que la herramienta se pueda llenar antes de la reunión por cada persona y luego comparar las diferencias, esto creo que ayudaría a enfocarse en los diferentes puntos de vista donde no se tiene un acuerdo previo.

6. Decisiones de Arquitectura

Proveedor de servicio de la nube

Descripción de la decision: selección del proveedor de servicio de la nube para la primera versión de la aplicación de la experiencia interactiva.

Integrantes:

- Juan González
- Diego Godínez
- Leo Godínez

Razonamiento de la Decisión:

Preocupación/Fuerza	AWS	Azure	GCP
Conocimiento Interno	Alto	Medio	Bajo
Serverless	Soporta	Soporta	Soporta
Soporte de Comunidad / Documentación	Alto	Alto	Medio

Tipo de Arquitectura de Software

Descripción de la decision: selección del tipo de arquitectura a usar para la aplicación de la experiencia interactiva.

Integrantes:

- Juan González
- Diego Godínez
- Leo Godínez

Razonamiento de la Decisión:

Preocupación/Fuerza	Arquitectura en Capas	Microservicios	Arquitectura Orientada a Servicios
Conocimiento Interno	Medio	Alto	Medio
Flexibilidad	Medio	Alto	Alto
Tipo de Aplicación	Soporta	Soporta	Soporta
Complejidad	Dependiente del contexto	Dependiente del contexto	Dependiente del contexto

Tipo de Tecnología

Descripción de la decision: selección del tipo de arquitectura a usar para la aplicación de la experiencia interactiva.

Integrantes:

- Juan González
- Diego Godínez
- Leo Godínez

Razonamiento de la Decisión:

Preocupación/Fuerza	Serverless	Contenedores	Servidores (Máquina Virtuales)
Conocimiento Interno	Alto	Medio	Alto
Tiempo de Desarrollo	Bajo	Medio	Medio
Mantenimiento	Medio	Medio	Alto
Portabilidad	Variable	Medio	Alto

Framework de desarrollo móvil

Descripción de la decision: selección del framework para el desarrollo de la aplicación móvil para la aplicación de experiencia interactiva

Integrantes:

- Juan González
- Diego Godínez
- Leo Godínez
- Steven Mora

Razonamiento de la Decisión:

Preocupación/Fuerza	Flutter	React Native	Angular Ionic
Conocimiento Interno	Bajo	Medio	Medio
Tipo de Aplicación	Soporta	Soporta	Soporta
Compatibilidad con Web	No Soporta	Soporta	Soporta
Multiplataforma	Sí	Sí	Sí
Estructurado	No	No	Sí

Base de Datos

Descripción de la decision: selección del servicio y motor de base de datos de la aplicación móvil para la aplicación de experiencia interactiva

Integrantes:

- Juan González
- Leo Godínez
- Steven Mora

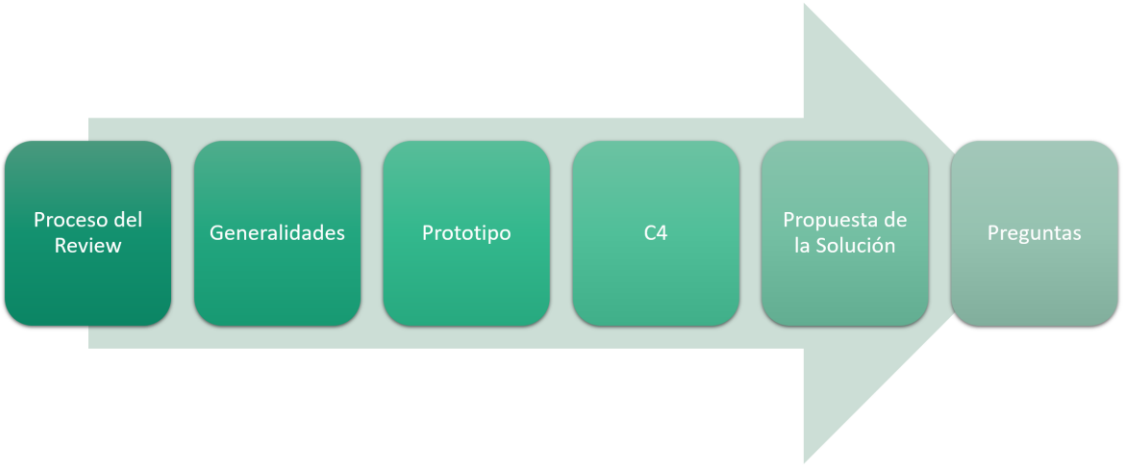
Razonamiento de la Decisión:

Preocupación/Fuerza	RDS - MySQL	Aurora Serverless	DynamoDB
Conocimiento Interno	Alto	Medio	Medio
Tipo de Aplicación	Soporta	Soporta	Soporta
SQL/NoSQL	SQL	SQL	NoSQL
Escalabilidad	Sí	Sí	Sí
Tecnología	DBaaS	Serverless	Serverless

7. Presentación de la revisión de arquitectura

Juan González

Agenda



Proceso del Review

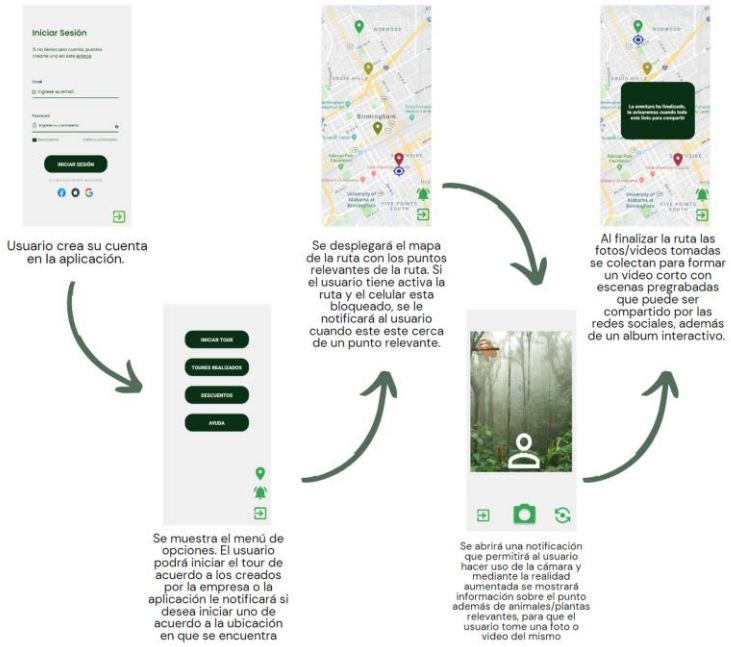
Generalidades

GodiTours

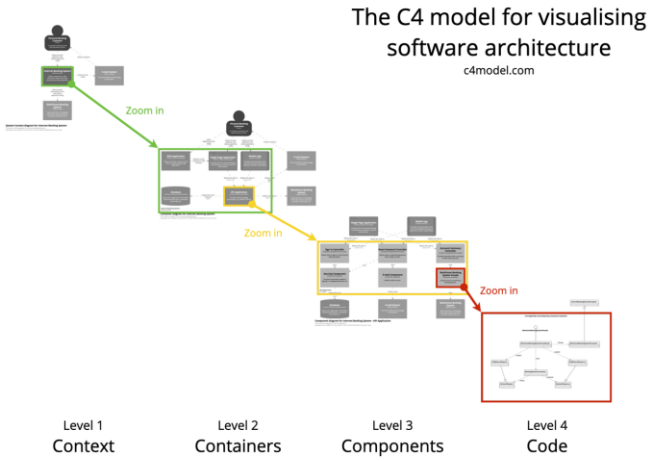
Experiencia Interactiva

Turismo Natural

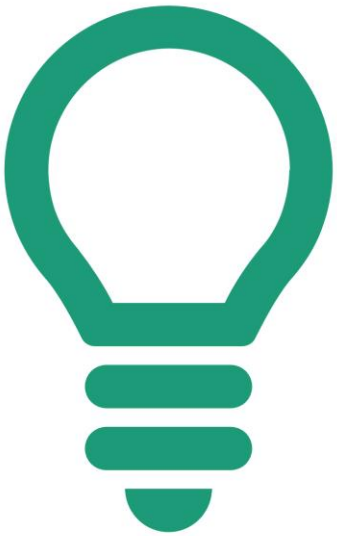
Prototipo



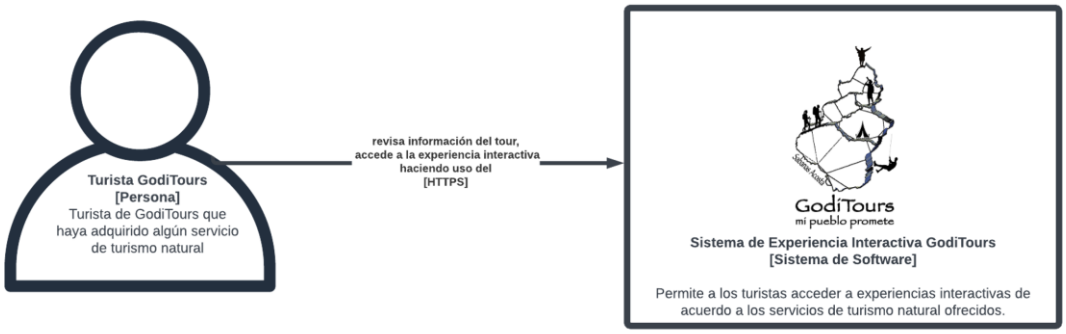
C4



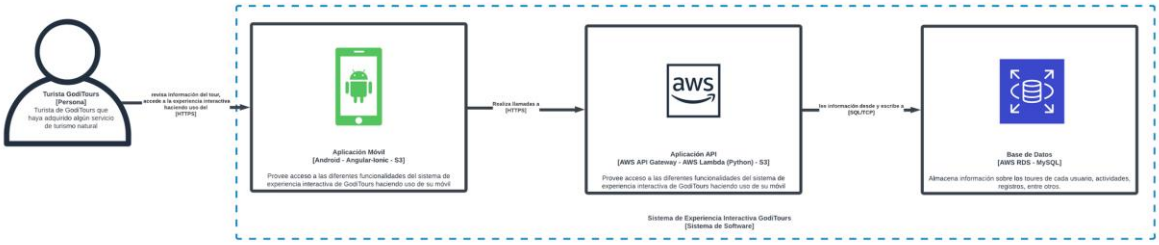
Propuesta de Solución

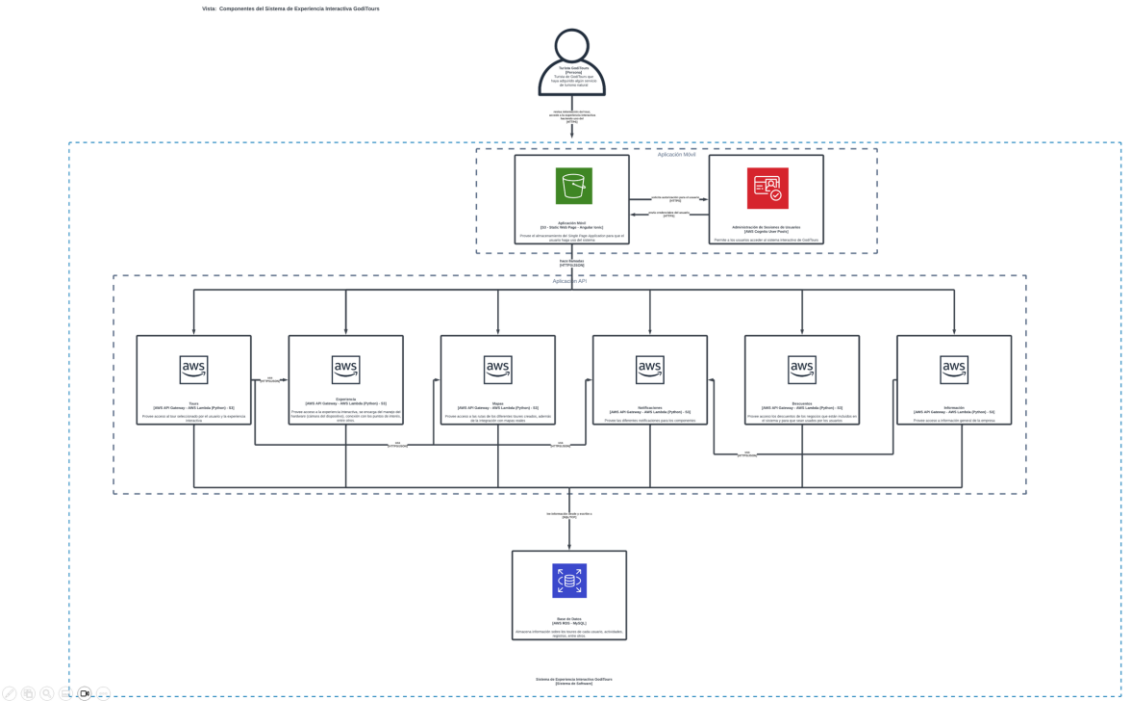


Vista: Contexto del Sistema de Experiencia Interactiva GodiTours

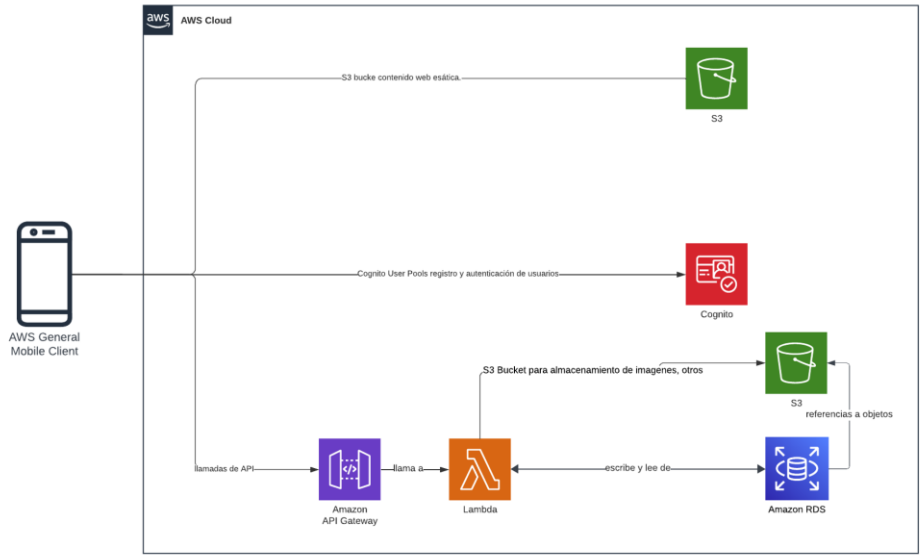


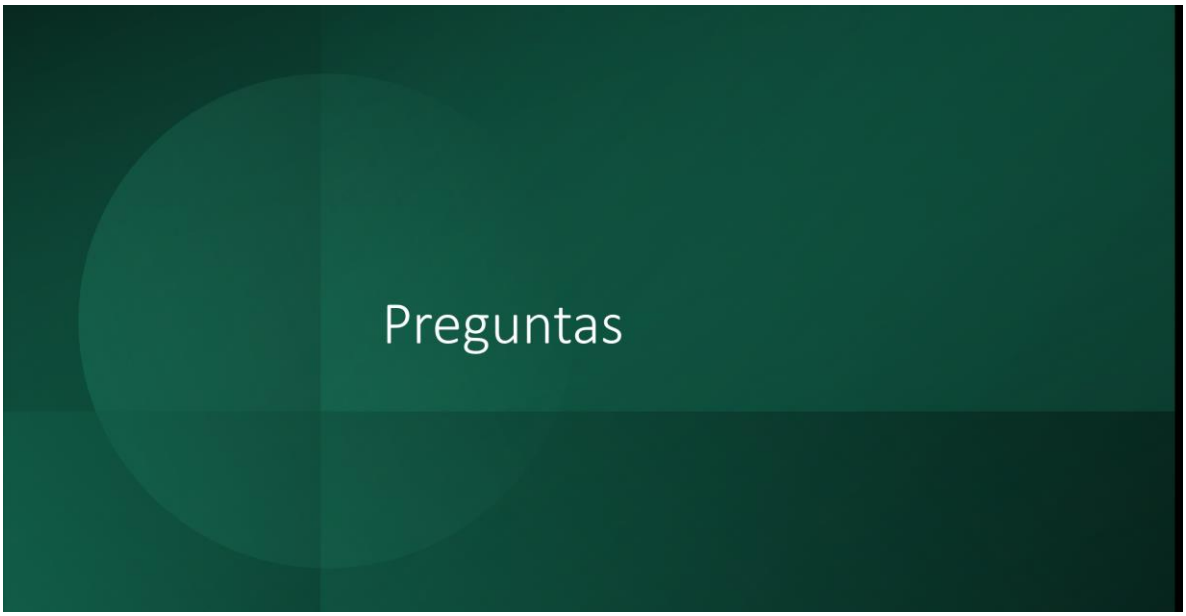
Vista: Contenedor del Sistema de Experiencia Interactiva GodiTours



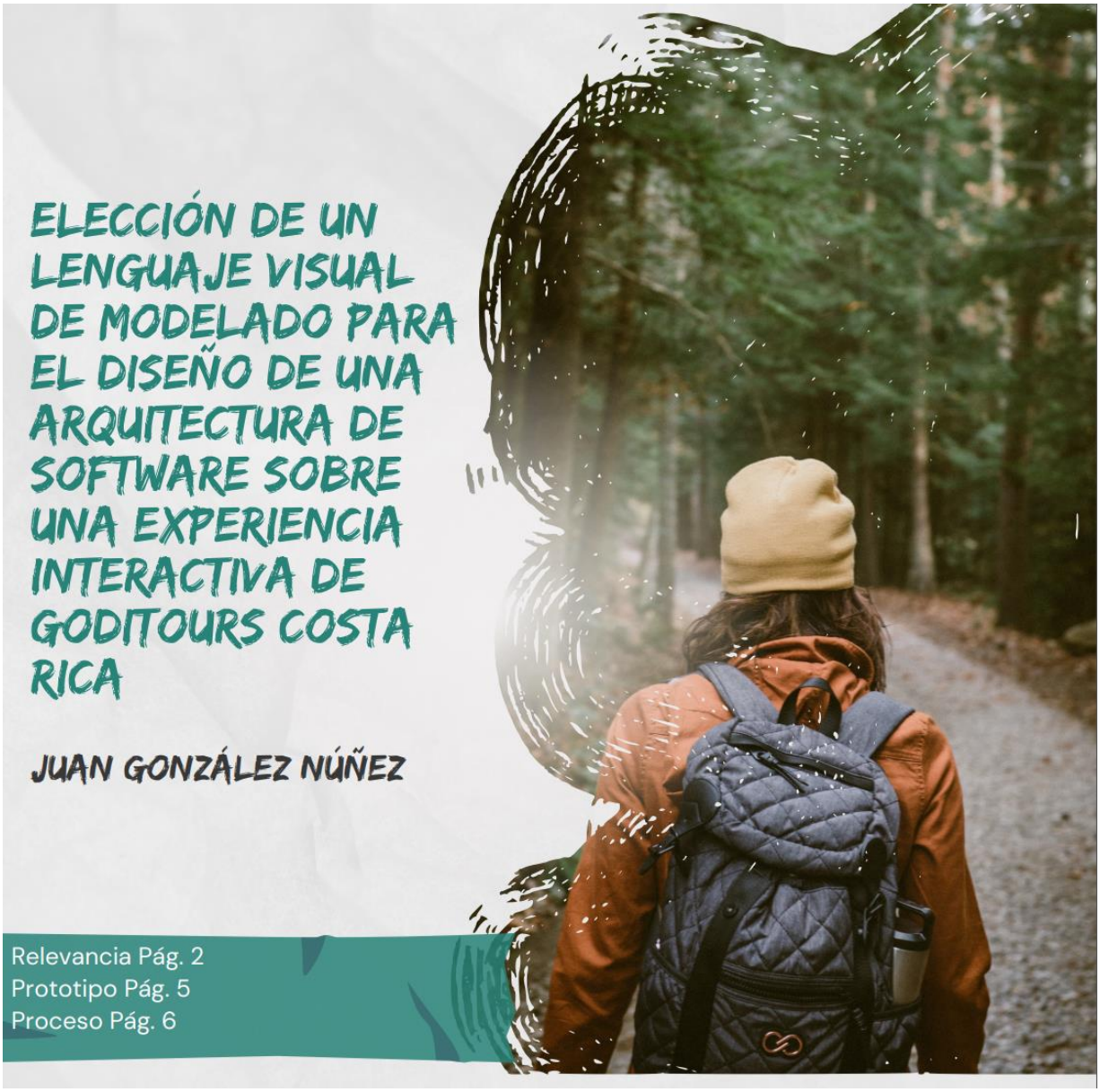


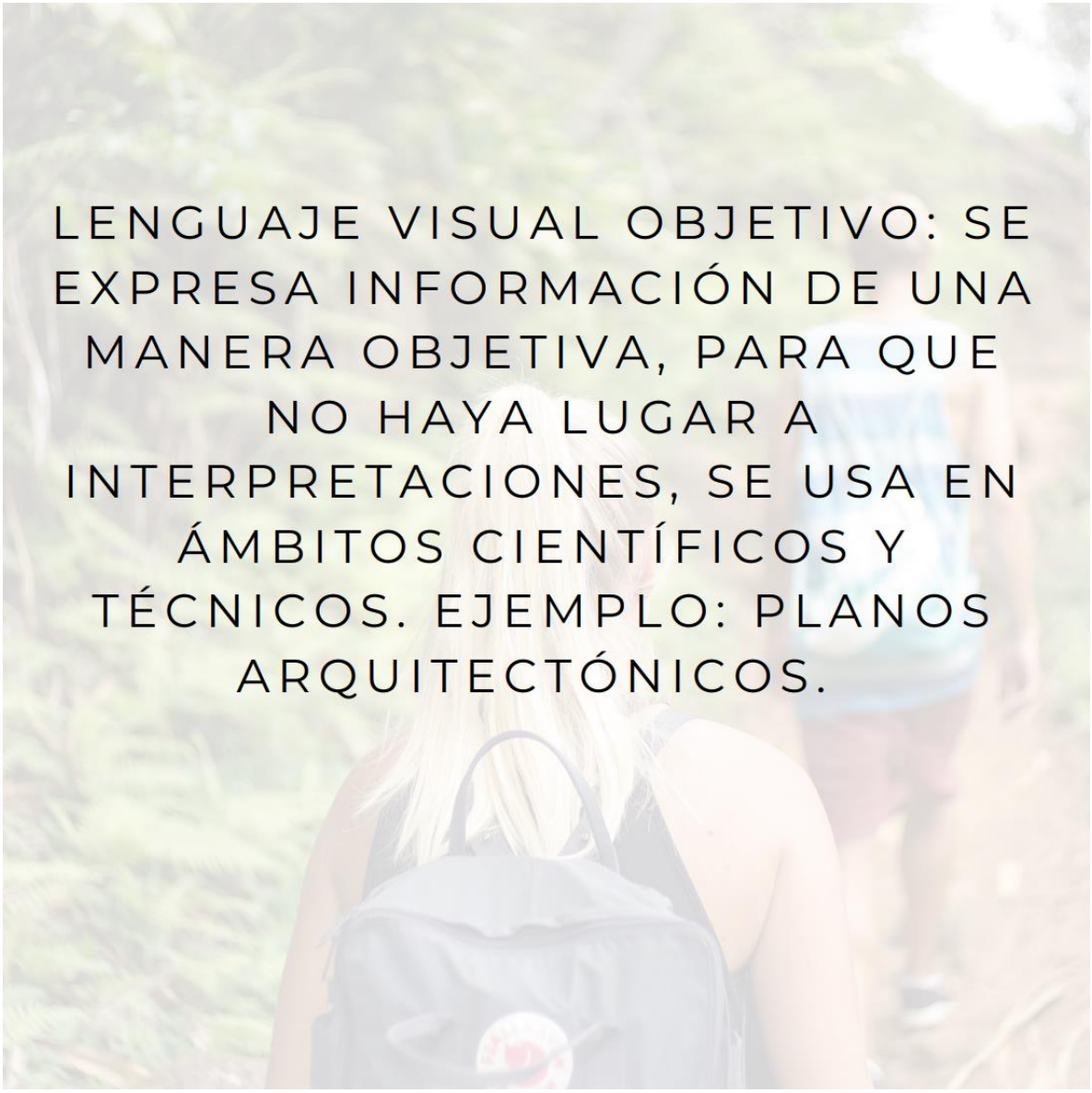
Vista: Código del Sistema de Experiencia Interactiva GodiTours




Preguntas

8. Documentación del contexto




LENGUAJE VISUAL OBJETIVO: SE EXPRESA INFORMACIÓN DE UNA MANERA OBJETIVA, PARA QUE NO HAYA LUGAR A INTERPRETACIONES, SE USA EN ÁMBITOS CIENTÍFICOS Y TÉCNICOS. EJEMPLO: PLANOS ARQUITECTÓNICOS.

Costa Rica es conocido a nivel mundial sobre su biodiversidad lo que hace que la atracción de turismo sea una de sus principales fuentes de ingreso. De acuerdo a la Oficina Económica y Comercial de España y Panamá esta industria genera un 8% del PIB, en otras palabras, un aproximado de 3.796,9 millones de dólares.

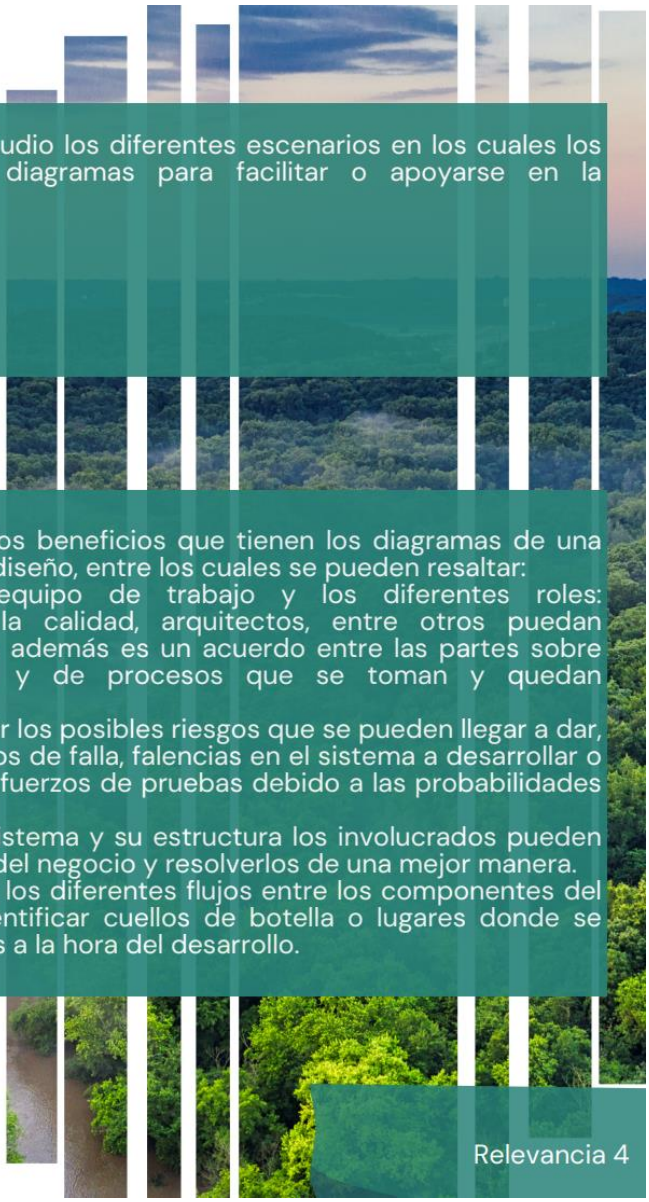
2 117 860 turistas ingresaron al país por todas las vías. ICT (2022)

Costa Rica es el tercer país latinoamericano en el índice de competitividad de viajes y turismo, por detrás de México y Brasil. Oficina Económica y Comercial de España y Panamá (2021)

La industria del turismo en Costa Rica genera un 28% de empleo directo e indirecto, siendo de los importantes pilares de la economía del país.

Relevancia 2


65% DE TURISTAS
REALIZARON
ACTIVIDADES DE
ECOTURISMO
ENTRE 2017 Y 2019
ICT (2020)



Cherubini et al. (2007) detalla en su estudio los diferentes escenarios en los cuales los ingenieros usan dibujos, bocetos o diagramas para facilitar o apoyarse en la comunicación:

- Comprender código existente
- Diseño
- Documentación
- Inducción
- Explicación a clientes

De acuerdo con AWS (s.f) son varios los beneficios que tienen los diagramas de una arquitectura de software en la etapa de diseño, entre los cuales se pueden resaltar:

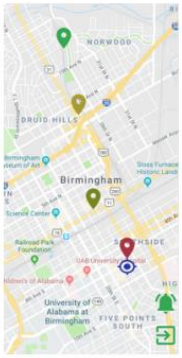
- Colaboración: ayuda a que el equipo de trabajo y los diferentes roles: desarrolladores, aseguradores de la calidad, arquitectos, entre otros puedan comprender el objetivo del negocio, además es un acuerdo entre las partes sobre decisiones a nivel de tecnología y de procesos que se toman y quedan documentadas.
- Reducción de riesgos: ayuda a reducir los posibles riesgos que se pueden llegar a dar, debido a que permite observar puntos de falla, falencias en el sistema a desarrollar o donde se deben enfocar mayores esfuerzos de pruebas debido a las probabilidades de fallas.
- Eficiencia: al permitir una vista del sistema y su estructura los involucrados pueden encontrar fallas en el entendimiento del negocio y resolverlos de una mejor manera.
- Escalabilidad: los diagramas, al tener los diferentes flujos entre los componentes del sistema, causan que se puedan identificar cuellos de botella o lugares donde se deben tomar en cuenta más variables a la hora del desarrollo.

Relevancia 4

Flujo de la Aplicación

Usuario crea su cuenta en la aplicación.

Se muestra el menú de opciones. El usuario podrá iniciar el tour de acuerdo a los creados por la empresa o la aplicación le notificará si desea iniciar uno de acuerdo a la ubicación en que se encuentra



Se desplegará el mapa de la ruta con los puntos relevantes de la ruta. Si el usuario tiene activa la ruta y el celular esta bloqueado, se le notificará al usuario cuando este este cerca de un punto relevante.

Se abrirá una notificación que permitirá al usuario hacer uso de la cámara y mediante la realidad aumentada se mostrará información sobre el punto además de animales/plantas relevantes, para que el usuario tome una foto o video del mismo

Al finalizar la ruta las fotos/videos tomadas se colectan para formar un video corto con escenas pregrabadas que puede ser compartido por las redes sociales, además de un album interactivo.

Prototipo 5


Proceso General de Investigación
Estudio de lenguajes visuales de modelado
Elección de un lenguaje visual de modelado
Diseño de arquitectura con el lenguaje visual de modelado
Proceso 6

Recomendaciones para el review

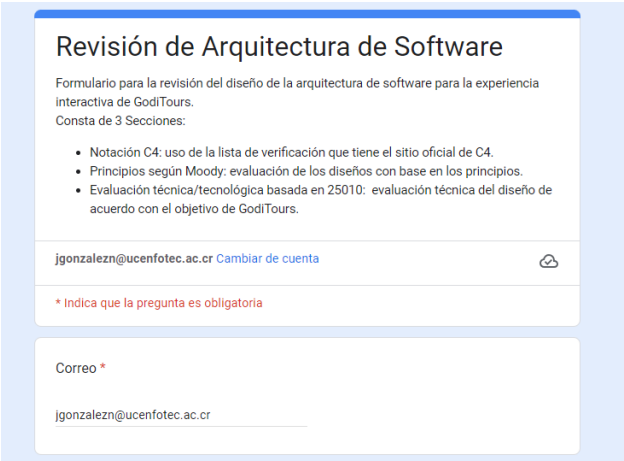
Principios de Moody

Principio	Descripción
Calidad Semiótica	Relación 1-1 entre los símbolos gráficos y constructores semánticos. Cada símbolo tiene un significado semántico
Discriminación Perceptual	Los símbolos deben de ser distinguibles entre sí.
Transparencia Semántica	El modelo debe tener mecanismos que permita ajustar la complejidad de acuerdo a la demanda. Modularidad, jerarquía, entre otros
Complejidad Administrativa	Si el proceso consiste en varios modelos, debería de haber maneras de incluir la información de otros modelos
Integración Cognitiva	Si el proceso consiste en varios modelos, debería de haber maneras de incluir la información de otros modelos
Expresividad Visual	Para incrementar la expresividad se deben de usar el rango completo de variables visuales. Por ejemplo: color, textura, entre otros
Codificación Visual	Modelo tiene que ser complementado con texto para dirigirse a ambos canales cognitivos.
Economía Gráfica	El número de símbolos debe estar limitado a un número cognitivo manejable
Ajuste Cognitivo	Dependiendo de las circunstancias el modelo debería ofrecer diferentes representaciones

ISO 25010: Características

Característica	Descripción
Adecuación Funcional	Representa la capacidad del producto software para proporcionar funciones que satisfacen las necesidades declaradas e implícitas, cuando el producto se usa en las condiciones especificadas
Eficiencia de Desempeño	Representa el desempeño relativo a la cantidad de recursos utilizados bajo determinadas condiciones
Compatibilidad	Capacidad de dos o más sistemas o componentes para intercambiar información y/o llevar a cabo sus funciones requeridas cuando comparten el mismo entorno hardware o software
Usabilidad	Capacidad del producto software para ser entendido, aprendido, usado y resultar atractivo para el usuario, cuando se usa bajo determinadas condiciones
Fiabilidad	Capacidad de un sistema o componente para desempeñar las funciones especificadas, cuando se usa bajo unas condiciones y periodo de tiempo determinados
Seguridad	Capacidad de protección de la información y los datos de manera que personas o sistemas no autorizados no puedan leerlos o modificarlos
Mantenibilidad	Representa la capacidad del producto software para ser modificado efectiva y eficientemente, debido a necesidades evolutivas, correctivas o perfectivas
Portabilidad	Capacidad del producto o componente de ser transferido de forma efectiva y eficiente de un entorno hardware, software, operacional o de utilización a otro

9. Formulario para la revisión de la arquitectura



Revisión de Arquitectura de Software

Formulario para la revisión del diseño de la arquitectura de software para la experiencia interactiva de GodiTours.
Consta de 3 Secciones:

- Notación C4: uso de la lista de verificación que tiene el sitio oficial de C4.
- Principios según Moody: evaluación de los diseños con base en los principios.
- Evaluación técnica/tecnológica basada en 25010: evaluación técnica del diseño de acuerdo con el objetivo de GodiTours.

jgonzalezn@ucenfotec.ac.cr Cambiar de cuenta

* Indica que la pregunta es obligatoria

Correo *

jgonzalezn@ucenfotec.ac.cr

Revisión de Arquitectura de Software

jgonzalezn@ucenfotec.ac.cr Cambiar de cuenta

Sección 1: Notación C4

¿Posee los diagramas un título?

- Sí
- No

¿Entiende de qué es el diagrama?

- Sí
- No

¿Entiende el alcance del diagrama?

- Sí
- No

¿Posee una descripción el diagrama?

- Sí
- No

¿Cada elemento tiene un nombre?

- Sí
- No

¿Se entiende el tipo de cada elemento?

- Sí
- No

¿Se entiende la tecnología asociada a cada elemeto?

- Sí
- No

¿Se entienden los acrónimos y abreviaciones usadas?

- Sí
- No

¿Entiende los colores usados?

- Sí
- No

¿Entiende las formas usadas?

- Sí
- No

¿Entiende el significado de todos los íconos usados?

- Sí
- No

¿Cada línea relacional posee una etiqueta que la describe?

- Sí
- No

¿Entiende el significado de cada punta de flecha usada?

- Sí
- No

Sección 2: Principios de Moody

Evalúe los principios de acuerdo con lo visto en los diseños y las definiciones *

	Alto	Medio	Bajo	No aplica
Calidad semiótica	○	○	○	○
Discriminación perceptual	○	○	○	○
Transparencia semántica	○	○	○	○
Complejidad administrativa	○	○	○	○
Integración cognitiva	○	○	○	○
Expresividad visual	○	○	○	○
Codificación dual	○	○	○	○
Economía gráfica	○	○	○	○
Ajuste cognitivo	○	○	○	○

Sección 3: Evaluación Técnica/Tecnológica basada en ISO 25010

Califique sus características de la calidad del producto de software, además de mencionar **deficiencia, omisión o mejora**

Adecuación funcional: *

Tu respuesta

Eficiencia de desempeño: *

Tu respuesta

Compatibilidad: *

Tu respuesta

Usabilidad: *

Tu respuesta

Fiabilidad: *

Tu respuesta

Seguridad: *

Tu respuesta

Mantenibilidad: *

Tu respuesta

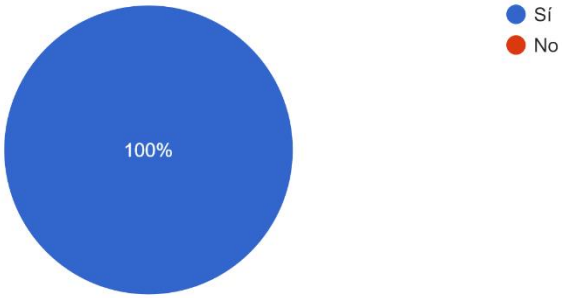
Portabilidad: *

Tu respuesta

10. Resultados del formulario para la revisión de la arquitectura

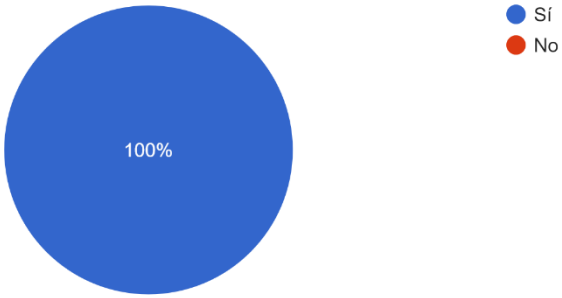
¿Posee los diagramas un título?

2 respuestas



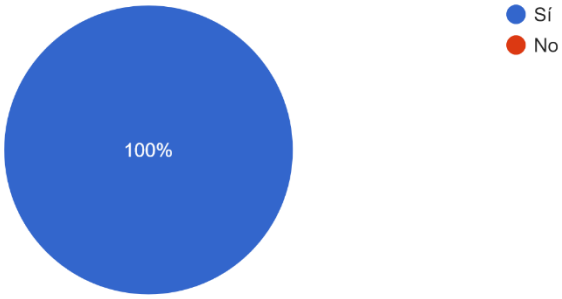
¿Entiende de qué es el diagrama?

2 respuestas



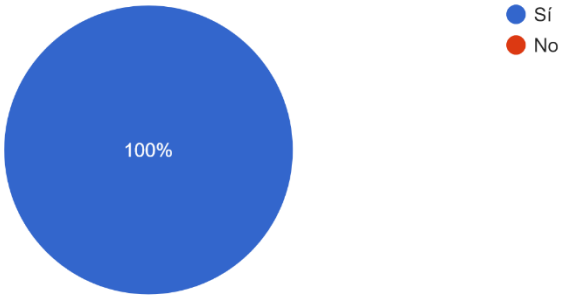
¿Entiende el alcance del diagrama?

2 respuestas



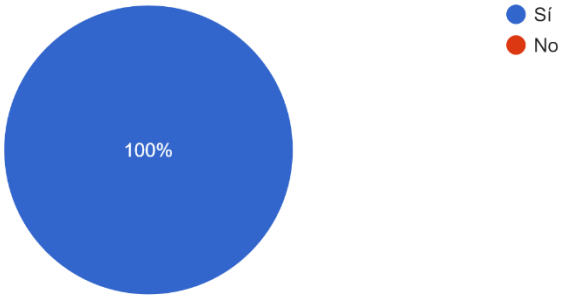
¿Posee una descripción el diagrama?

2 respuestas



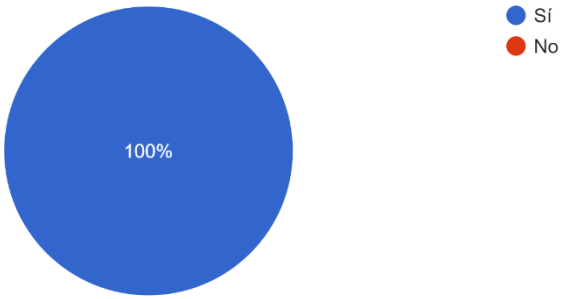
¿Cada elemento tiene un nombre?

2 respuestas



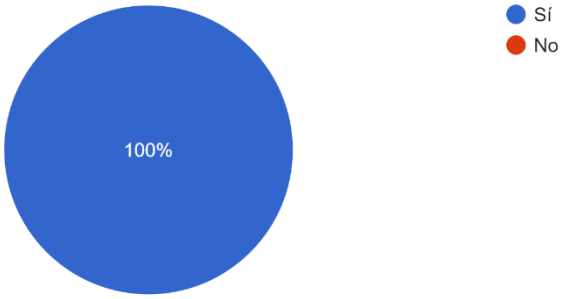
¿Se entiende el tipo de cada elemento?

2 respuestas



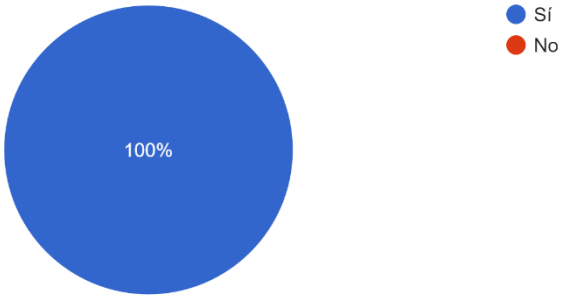
¿Se entiende la tecnología asociada a cada elemeto?

2 respuestas



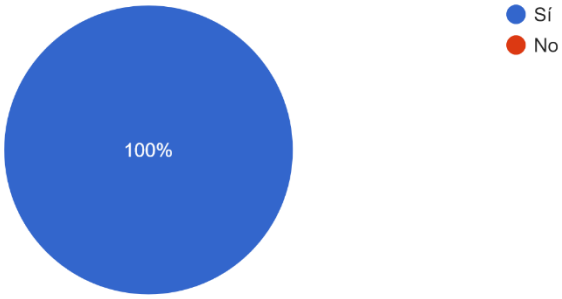
¿Se entienden los acrónimos y abreviaciones usadas?

2 respuestas



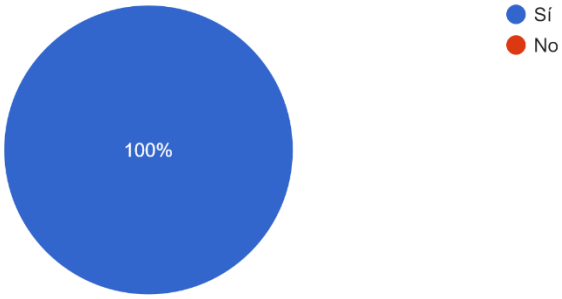
¿Entiende los colores usados?

2 respuestas



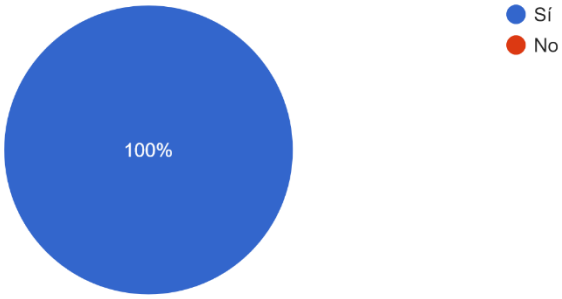
¿Entiende las formas usadas?

2 respuestas



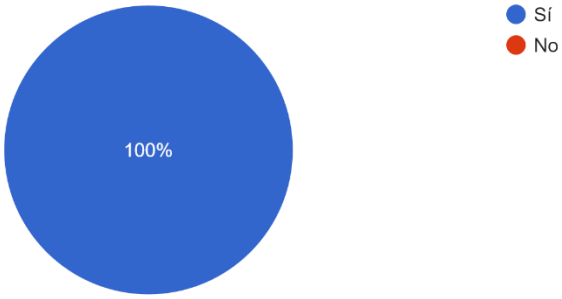
¿Entiende el significado de todos los íconos usados?

2 respuestas



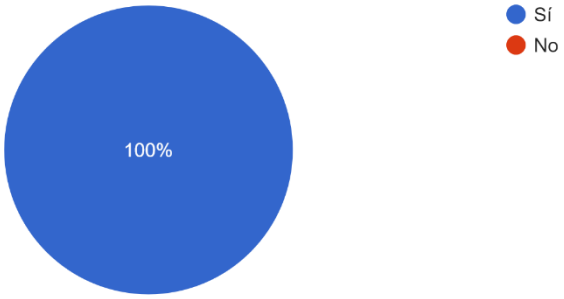
¿Cada línea relacional posee una etiqueta que la describe?

2 respuestas

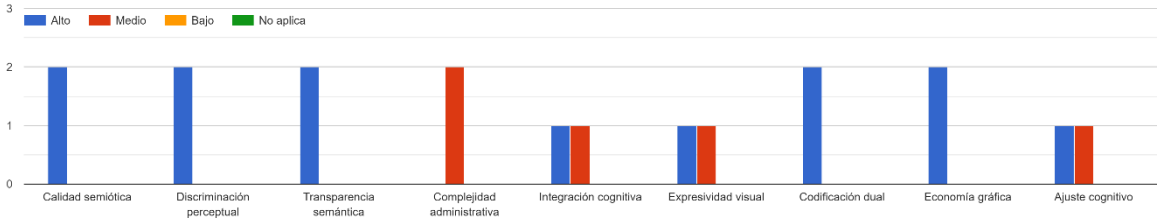


¿Entiende el significado de cada punta de flecha usada?

2 respuestas



Evalúe los principios de acuerdo con lo visto en los diseños y las definiciones



Adecuación funcional:

2 respuestas

Adecuada

Se considera que está bien

Eficiencia de desempeño:

2 respuestas

Mejora: El uso de cache para administrar ficheros de S3 frecuentemente accedidos. Evaluacion de CDN para reducir recursos. Tomar en cuenta que bajo el uso normal de la aplicacion, es posible que algunas rutas de tours se encuentren en lugares con baja cobertura de internet, lo cual puede impactar negativamente la experiencia de usuario.

Se considera que está bien

Compatibilidad:

2 respuestas

Aceptable. El unico "problema" es el uso de tecnologias propietarias de AWS lo cual puede generar problemas si se desea utilizar otra plataforma o recursos propios.

Se considera que está bien

Usabilidad:

2 respuestas

Adecuada. Puesto que es una aplicacion de Android, tomar en cuenta los distintivos tamanos de pantalla (desde celulares hasta tablets).

Se considera que está bien

Usabilidad:

2 respuestas

Adecuada. Puesto que es una aplicacion de Android, tomar en cuenta los distintivos tamanos de pantalla (desde celulares hasta tablets).

Se considera que está bien

Fiabilidad:

2 respuestas

Mejora: Redundancia de S3, puede la aplicacion continuar si hay perdida de la informacion de los buckets (caidas, corrupcion de data, perdida de permisos, etc).

Se considera que está bien

Seguridad:

2 respuestas

Aceptable. Uso de HTTPs en la capa de aplicacion junto a cognito ofrecen una barrera aceptable para informacion en transito y almacenada. Validar las configuracions de permisos, IAM, firewall, etc a nivel de aplicaciones de AWS.

Se considera que está bien

Mantenibilidad:

2 respuestas

Aceptable. Verificar el diseno de dominio de las APIs si poseen flexibilidad para ser actualizadas sin necesidad de romper los contratos (Open-Closed principle).

Se considera que está bien

Portabilidad:

2 respuestas

Aceptable. El uso de REST APIs permiten en un futuro consumir las funcionalidades desde otras plataformas

Se considera que está bien